A Framework for Multiple Adaptable Pedagogical Strategies in Intelligent Tutoring Systems

---

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Doctor of Philosophy
by
Moffat Mathews

Supervising Committee

Senior Supervisor :	Professor Dr. Antonija Mitrović
Associate Supervisor :	Dr. Michael Grimley
Associate Supervisor :	Dr. Brent Martin

Examining Committee

Prof. Dr. Jim Greer	External Examiner
Prof. Dr. Judy Kay	Internal Examiner
Prof. Dr. Antonija Mitrović	UC Examiner

University of Canterbury
2012

To my mum and dad,
who although are gone
are never far away.

Abstract

The need to give educators the ability to enter a particular teaching strategy of their choice into an Intelligent Tutoring System (ITS) and have the ITS respond appropriately to each student has been stated by many researchers. For example, an educator could tell the ITS to keep students within a certain help level ratio (how much help they request), or to introduce a new topic in a particular manner and the ITS simply carries this out at each learning point of interest. Educators could then try new strategies, ones that unaided are impossible to try out in class (such as keeping a student within a help-seeking range) or difficult within an ITS (as the ITS would have to be specially programmed in that way). Current ITSs provide adaptivity to the student at the domain level but not necessarily at the pedagogical level. While a variety of pedagogical strategies have been implemented (e.g. apprenticeship, socratic, practice), there is no system that offers parts or all of these strategies with the ability to choose between them dynamically.

In this project, we designed a new framework for an ITS to include multiple, potentially adaptable pedagogical strategies. This was done by breaking up the pedagogical module into separate components. The Pedagogical Strategy Set (PSS) contains all the strategies, written as constraints. The Pedagogical Student Model (PSM) keeps track of which pedagogical strategies were used by each student. Within the ITS, there is still a smaller, separate pedagogical module to deal with domain-specific strategies. The Pedagogical Control Centre (PCC) contains the logic of when and how to use the pedagogical strategies. It gathers its information from the other modules and uses decision logic to trigger strategies.

We implemented and evaluated this framework within the context of SQL-Tutor and found that the framework could be used to enter pedagogical strategies, which in turn compared favourably to the original SQL-Tutor. This proof of concept opens up the possibility of the logic and algorithms that could be implemented (e.g. in the PCC) in future ITSs. The PSS is a separate module, written in a different language, independent of ITSs. This could lead to sharing of pedagogical strategies between tutors. Furthermore, students learn differently to each other; this framework allows them to do so.

Table of Contents

List of Tables

List of Figures

Acknowledgments

During this project, life, as it always does, went on. I had some great times, but I also had some very stressful times (unrelated to the project), some lasting for quite a long time. On top of all that, my dear dad died unexpectedly. All of this makes what I am about to say so much more than it would have been, had life been more "normal".

I would like to thank all the members of my supervisory team for their comments, advice, and encouragement along the way. In particular, I am grateful to Dr. Antonija Mitrović for being my primary supervisor. The ideas in this project are intertwined with her thoughts, her guidance, and her supervision. Tanja listened to whatever I had to say, even the crazy ideas, and guided me without dulling my enthusiasm. She believed in my ability to complete the project and to carry out my teaching and research workload as well. Her holistic approach meant that she not only knew about my project, but also about things in my personal life that had the potential to affect me as a person. This meant that her caring extended far beyond that of a supervisor. Tanja, thank you for believing in me.

Many thanks go to my research group, the Intelligent Computer Tutoring Group (ICTG) for the constant encouragement, support, and for keeping the exchange of ideas always going; challenging and refreshing my mind. In particular, I would like to thank Jay Holland and Amali Weerasinghe for many in-depth conversations, arguments, and scrutiny. The camaraderie within ICTG has been such that the group has become my extended family. There were also those who critiqued and challenged my ideas and work in a very caring manner. I would like to thank both Prof. Benedict du Boulay and Prof. Gordon McCalla for spending time trying to comprehend and make sense of my ideas and for their valuable contributions to this endeavour. Thank you all for making me a better researcher.

I have a bad habit of taking on projects and tasks even when my plate is full, particularly thankless, time-consuming tasks. Peter Glassenbury noticed

this early on in my PhD, and hoping I (and others) would 'get' it, aptly named me the *department mug*[1]. He even presented me with an actual mug to sit on my shelf as a constant reminder not to undersell myself. During the write-up, he vigilantly kept protecting me from this (from myself!). He has without fail checked up on me every time he saw me, and always had a word of encouragement. Thanks Pete for the tough love presented in the most gentle manner! I appreciate that.

I was lucky to have done my thesis in a very supportive department: the Computer Science and Software Engineering Department at the University of Canterbury. Every staff member of the department has at some stage made my life easier, pulled me out of a slump, looked out for me, or made me smile on a day when smiles were scarce. The administration staff who made everything easy for me (thanks Gillian Clinton and Alex Forster!). Yalini Sundralingam who was always proud and a great supporter. Neville Churcher and Tim Bell, who looked out for me with my teaching course load. Adrian White, who took me on artistic journeys with our photography interests; not forgetting the philosophical ones too! To Joffre Horlor for answering the long list of my work-based technical questions, saving me time to do my PhD. Phil Holland for always keeping all our machinery working and in top order. The list goes on.

There is one family, the Irwins, who with their closeness and kindness helped me hugely, especially during our stressful period. Wal, Liz, Isaac, and Riley, thank you for being there when I needed you. Thank you for a non-judgemental relationship filled with a huge amount of caring.

Last but not least, I want to thank the person who has stood by me through all of this, and has never known me without the burden of my project: my partner, Stephanie DeMay. Stephanie took care of so many of the things I could not, so that I could devote more time to my work and study. She also spent countless hours helping me proof read this thesis. Her unfaltering love and belief in me made me believe in myself. Stephanie, thank you for making me a better person.

This work was partially funded by the Top Achiever Doctoral Scholarship

[1] mug: a doormat, a soft touch, sucker, easy victim

from the New Zealand Tertiary Education Commission.

Thank you.

Chapter I

Introduction

One-to-one human tutoring (i.e. one student to one teacher) is held as the gold standard in tutoring [36]. In this situation, the teacher knows both the domain and the student well and therefore can make good pedagogical decisions at the appropriate time to benefit the student. Knowing the student means that the teacher has a fairly precise idea of the student's knowledge in each part of the domain. The teacher also has a history of the student's progress, and therefore can provide appropriate feedback. For example, if a student has, after many unsuccessful attempts, successfully learnt the concept, the teacher might give more praise than if the student had learnt the concept straight away. In another example, a teacher could use analogies to link similar concepts together, knowing that the student has already successfully learnt the first concept to which the link is being made. As we get further away from this student-to-teacher ratio, the ease and efficiency of teaching decreases. Holding the knowledge of multiple students and trying to customise the learning process for each student quickly drops off as a teacher takes on more students. Unfortunately, most classrooms have a high student-to-teacher ratio. In schools, this might be typically one teacher to around 30 students. At tertiary levels (particularly in the undergraduate courses), this ratio could be significantly greater, even in tutorials where students are practising what they have learnt in lectures. This means that students are not learning as effectively as they could be. Providing each student with a learning environment where he/she can practice what they have learnt, while receiving customised feedback, would go a long way to solving this problem.

Intelligent Tutoring Systems (ITSs) are computer-based learning environments that are intelligent. This means that, unlike standard e-learning systems, they "know" about the domain and "know" about each student

(i.e. by keeping a model of each student with respect to the domain), so that they can reason about a student's knowledge in that domain. This means that ITSs know facts about the domain to such an extent that they can make reasonable pedagogical judgements for each student at the various learning points as the student progresses through the learning session. The system holds information about each student's knowledge at the level of the domain facts or problem steps and thus is able to customise the way it tutors each student. Each student, therefore, travels a different path through the system. As an example, some tutoring systems are able to suggest the "next best topic or problem" that the student should attempt. This decision would be based on each student's model, and therefore would be customised to that particular student. This project deals with ITSs; more specifically, the pedagogical decisions made by ITSs.

1.1 The Problem

Good human tutors have several pedagogical strategies that they can use at any one learning point; ITSs usually do not. Human tutors can choose a particular strategy from this set of strategies to use with a particular student for that particular learning point. Furthermore, they can adapt and change strategies depending on the student's behaviour with a particular strategy (i.e. not all strategies might work well with each student). Due to the difficulty in implementation, most ITSs assume one strategy at each learning point. This is also not adaptive to the student.

For example, when introducing the student to a new topic, a human tutor could: give them a verbal or textual tutorial about it; show them a worked example; give them a problem and then provide explanations as the student reaches impasses; use a socratic method of inducing the topic and its relevance; or show them a problem and use inquiry-based learning to let the student drive the learning process. All of these strategies are for that one learning point of introducing a new topic. The human tutor then has to make a decision to use one of these pedagogical strategies. This decision could be based on what strategy worked (or did not work) previously, the student's knowledge of the domain, any implications from the domain, and

quite often, trial and error. The time between monitoring and making these decisions might vary, so the student's performance on that decision needs to be recorded somewhere (in the case of human tutors, often mentally).

Authors of ITSs look to good human tutors and educational research for guidance on pedagogical decisions within systems that they build. Researchers and developers of ITSs have long wanted the ability for ITSs to have a choice in the range of pedagogical strategies for each learning point a student encounters. However, since this is a very difficult task to implement, most ITSs have a fixed pedagogical strategy for each point of learning. For example, they might have one method for choosing the next best topic or problem for the student; the student might be given the ability to overide this decision. Another example is the amount of help a student uses within a system. Many ITSs do not have a strategy to cope with this, and leave the decision to the student, which knowingly or otherwise is an underspecified strategy in itself. Some students end up using too much or too little help and this can lead to unwanted behaviours, such as when students game the system. ITS developers also have to work within the constraints of a computer-based application (for example, difficulties with natural language). This compounds the difficulty of implementation.

In the early days of ITSs (and even CAIs), authors of these systems researched into instructional planning to understand how to best teach the student. Instructional planning is divided into content planning and delivery planning [177]. Content planning defines the content the student should see at a given time. Delivery planning specifies how and when the content should be delivered (i.e. pedagogical rules). Content was first specified with the Learning Objects Model (LOM), where learning objects were reusable chunks of content. Learning outcomes were then defined. For example, the learning outcomes in Gagné's taxonomy are intellectual skills, cognitive strategies, verbal information, attitudes, and motor skills [127]. The next step was to create ontologies, where learning objects were connected together to form learning resources that could be shared amongst tutoring systems. It was important that these ontologies were standardised to allow for sharing and reusability. A number of markup languages were created to describe these ontologies, such as Learning Object Markup Language (LOML) and

Learning Material Markup Language (LMML). These described the structure of tutorials. For example, the LOML has an "example" node where an example of the particular concept is held. It also contains a "test" node, which contains a test to examine the student's knowledge of that concept. Pedagogic knowledge in these systems was usually described in the form of rules. The rules were generally independent of content, allowing the same instructional planner to handle content from many different domains. Using this, content and delivery could be tailored to students [128].

Another important step in describing the modelling of learning processes was the creation of the Educational Modelling Language (EML) developed at the Open University of the Netherlands. This has now been superseded by the IMS-Learning Designs (IMS LD), which is a specification maintained by the IMS Global Learning Consortium[1] [76].

The IMS-LD is pedagogically neutral (it does not adhere to any pedagogical theory). The specification is similar to a play. People act in different roles. Roles support certain activities. Activities are conducted within an environment that contains learning objects and services. IMS-LD is written in XML and requires an IMS-LD aware tool to play a unit of learning. This is still used today, primarily in LMS-type architectures. Today's ITSs have complex domain and student models (which the ITS uses to reason about the student's knowledge). Furthermore, pedagogical strategies are simply not limited to just certain constructs (e.g. example, test, sequencing), but are almost limitless in research.

Most ITSs nowadays have used some variation of the standard architecture proposed which consists of four, sometimes five, modules. These are the domain knowledge model, the student model, the communications module, and the pedagogical module and sometimes a fifth component, the expert model, is also included. Variations of this architecture have worked very well so far and have given us many different types of ITSs that have been both useable and effective. However, the fact still remains that, if an ITS author wants to implement a particular pedagogical strategy, they have to change (or create) the tutoring system. Usually, this involves quite a large change

[1] http://www.imsglobal.org/learningdesign/

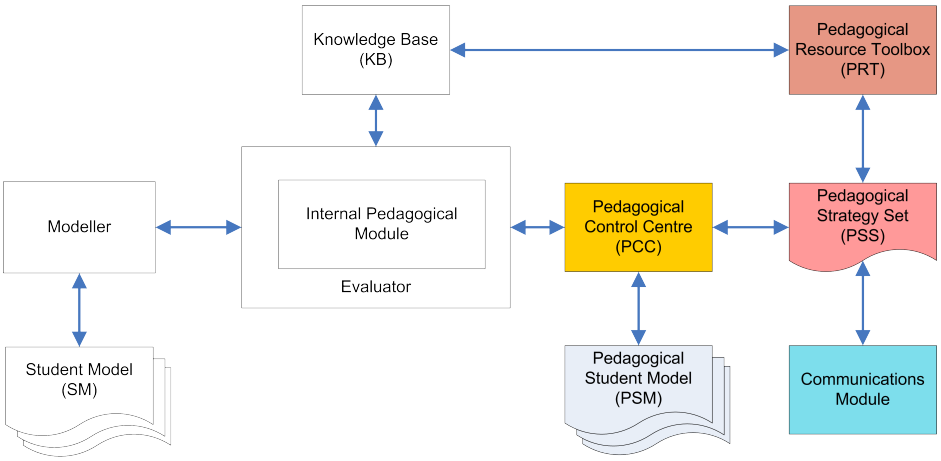

Figure 1.1: Architecture of the evolved MAPS Framework

to the underlying code. Quite often, the new strategy replaces the existing strategy rather than being another option for adaptation. Furthermore, it does not allow for strategy re-use in another tutoring system. What is required is a new framework for building ITSs, where multiple strategies could be implemented at once.

1.2 A Possible Solution

This project looks at the problem of the current ITS architectures allowing for only one pedagogical strategy at each point of learning and attempts to design a new framework which allows for multiple, adaptable pedagogical strategies (MAPS) for building ITSs. We have called this the MAPS Framework. The final design of the MAPS Framework architecture can be seen in Fig. 8.3, shown again here in Fig. 1.1.

In this framework, there is still a pedagogical module internal to the ITS that takes care of domain-specific pedagogical decisions. A Pedagogical Strategy Set (PSS) contains all the domain-independent strategies. The Pedagogical Resource Toolbox (PRT) contains any resources that might be used by the tutoring system; examples of these are problems, worked examples, and tutorial videos. The PRT is connected to the PSS as different strategies would need to use the various available resources. At the centre of the pedagogical decision making is the Pedagogical Control Centre (PCC).

This contains the logic that drives which strategy to use. Each student has a Pedagogical Student Model as well as their normal student model, all of which also inform the PCC. Sometimes strategies call for different graphical user interfaces. These are stored in the communications module; the strategies in the PSS can request the need for a different user interface to the communications module.

We started our design process with the first design of the MAPS Framework in Fig. 4.8. We used datamining and evaluation studies to guide our process to create and use the final design. This thesis is written as a historical process from the initial design to the final design. We also have a Chapter (Chapter 9) where we look at potential uses beyond the Framework design; some of which we have already implemented.

For the implementation and studies, we used SQL-Tutor, a constraint-based ITS. As our design does not deal with domain modelling, this design should in theory be able to be used with any tutor, regardless of the method of domain modelling. However, we have not tried this with non-CBM tutors within the scope of this project.

1.3 Guide to the Thesis

Chapter 2 and 3 set the scene by introducing the current context in the form of background information. After these two Chapters, we delve into the project work itself. Chapter 4 looks at the formation of a strategy using data mining techniques. We propose and build the first version of the MAPS Framework here, backed by data from the study and the background information. In each of the following chapters, we build on our design of the Framework as we gain more knowledge and insight into both learning and teaching, and taking into account implementation issues. In Chapter 5, we look at another data mining study. Here we see how students learn in a typical constraint-based ITS. We also make our decision to have the pedagogical layer independent of any particular domain modelling method. In Chapter 6, we build on our knowledge and use the MAPS Framework with a complex strategy, but with a human tutor carrying out some parts of the study. This is to check whether our design is feasible before spend-

ing a vast amount of effort devoted to implementing the new version of the Framework. With the success in Chapter 6, we implement the Framework with the complex strategy in Chapter 7. This gives us our proof of concept that the Framework works in this (and similar) situations. Chapter 8 looks at instances when a strategy calls for a user interface that is different to the standard tutor interface. This is a complex situation. In this chapter, we add this ability to the Framework design, implement it, and use it in a study. We then look at situations that this Framework could be used, beyond this design in Chapter 9. This is similar to a "future work" section, except that parts of the proposed future work have already been implemented. We then finish up with conclusions to the project.

Chapter II

Intelligent Tutoring Systems

Using computers to provide enhanced educational support to students has been employed since the late 1970s. A lot of the earlier work on computer based systems was founded on a desire to build the ultimate educational tool that acted like a good human tutor, incorporating the correct pedagogies and behaviours [42]. The earliest systems were called Computer Based Tutoring (CBT) or Computer Aided Instruction (CAI). Most of these tutors presented static material to students, where every student was presented with the same material. Some systems allowed students to have navigational control to view any content they wished while others forced strict adherence to the order of the material. Since then, much work has been done on developing these systems into three overlapping types of e-learning systems (see Fig. 2.1).

The first type is very similar to the original CBTs but with extra features. These are the Virtual Learning Environments of today, of which Learning Management Systems (LMSs) are the most popular. Today's LMSs allow for domain content (which the student can access in any order), forums, links to web pages and attachments, and quizzes. In the background, usually a

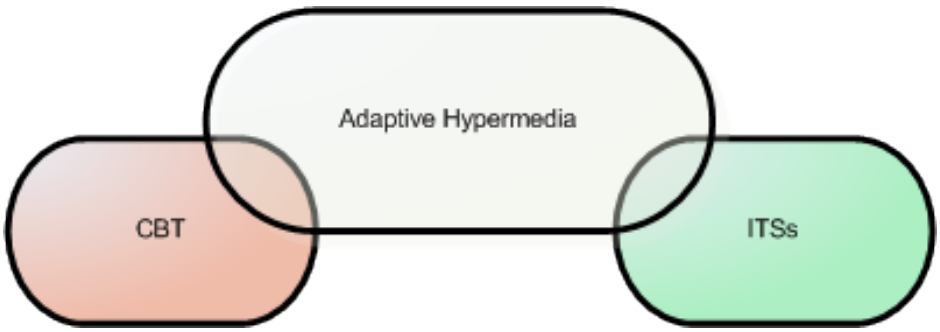

Figure 2.1: The relationship between CBT, AH, and ITSs

lot of data is being collected, so that the teacher can see who accessed which parts of the system and the time of access. The content is organised by the teacher as web pages or documents; the system itself has no understanding of the content or the structure of the domain. This type of system promotes discovery learning. It does not matter where the student starts or what path the student takes through the system. The path itself might not be connected or organised in any logical manner.

The second type are Adaptive Hypermedia or Adaptive Multimedia systems. This means that the system adapts in some way to either the student or the course content or both. An adaptive system might adapt the navigation based on some heuristics (e.g. disabling a link because the student has not covered the pre-requisite module). Navigation based adaptation is given as: direct guidance (e.g. a "next" button); adaptive sorting (re-sorting the links of a document according to what the user has seen before); adaptive hiding (hiding certain irrelevant or distracting links); and link annotation (giving extra information by adding colour or icons to a link) [38]. An adaptive system may be bound to a specific set of concepts. Concept-based systems use a domain model or content model to structure the information [16]. In these systems, adaptation is based not only on the user's preferences but also on what the system thinks of the student's current knowledge state.

The third type are ITSs. ITSs are learning systems that customise their feedback to the individual student by dynamically reasoning about the student's knowledge [30]. ITSs were created with a hope to get closer to the gold standard of learning, obtained by one-to-one tutoring. ITSs have a domain model which structures the domain and concepts in a way that the computer can *understand* it. ITSs also keep a student model for each student. This is the system's view of the student's knowledge about these concepts; it gains this from the student's interactions with the system. Using both the student and domain models, it can reason about the knowledge state of the student in that domain. Short-term student models can provide appropriate guidance on the specific task or error, while long-term (historical) models can play a part in adaptive pedagogical strategies (e.g. presenting the student with the next appropriate concept or problem).

A few of the CBTs are mildly adaptive. One fairly recent example of this

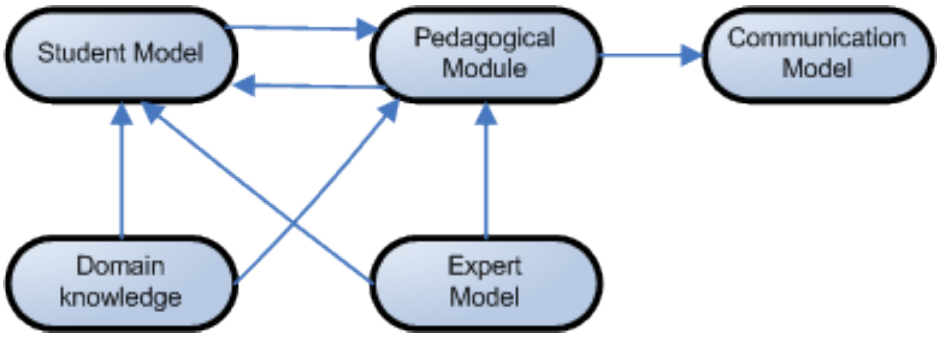

Figure 2.2: The interactions between components in an ITS [30]

is the *Lesson Module*[1] in an LMS called Moodle (Modular Object-Oriented Dynamic Learning Environment)[2]. The Lesson Module is based on the use of "pages". There are two types of pages: content pages and question pages. Teachers put their content (which could be text, HTML, multimedia, etc.) on content pages. The content pages are structured in such a way that they have a button at the bottom of the page that takes you to the next page. Question pages introduce branching into this process. Question pages contain a question with a number of possible answers. Choosing a particular answer could take you to an entirely different page; this could occur with both correct and incorrect answers. Using content and question pages appropriately could see students taking different paths through the knowledge base depending on their choices (i.e. their knowledge).

Each of these types of systems have a default pedagogy which is built into the system. We will be concentrating on ITSs in this project.

2.1 Intelligent Tutoring Systems Architecture

The structure of an ITS is typically divided into four main components: the domain knowledge model, the student model, the communications module, and the pedagogical module [187, 141]. Often a fifth component, the expert model, is also included [30]. Fig 2.2 shows the interactions between the components. Most ITSs are built using some version of this basic structure.

[1] `http://docs.moodle.org/22/en/Lesson_module`

[2] `http://moodle.org`

2.1.1 The Domain Module

The Domain Module contains the domain knowledge model. This module contains domain knowledge (facts and rules about the domain) represented in such a way that the ITS can *understand* it and use it to reason about the student's knowledge. This module can contain domain knowledge as procedural rules, constraints, or frames (pages) of knowledge content. For example, in Constraint-Based Modelling (CBM) tutors, this contains all the constraints that form the knowledge base of the tutor.

Earlier domain modules were classed as black box modules, where although they were able to provide solutions to problems, they could not describe the rationale behind them. More recent ITSs have domain modules that not only are able to provide the solution to a problem, but provide in depth explanations to the solution process. For example, Model Tracing (MT) tutors have all the correct and incorrect steps of a solution mapped out with very specific feedback attached to each step. This means that if the student makes an error on one step, they are not only told that they have made an error but are presented with various levels of feedback. In a CBM tutor, each constraint describes one domain principle. If the student makes an error, it means that one or more constraints have been violated. Each constraint can have multiple levels of feedback associated with it. When an error has been made, students not only know they have made a mistake, but can get varying levels of feedback based specifically on their error.

2.1.2 The Student Modeller

The job of the student modeller is to evaluate a student's solution and also maintain a model of that student (the student model). The student model is a representation of the student's knowledge and skill level within that domain. It should contain all the student's strengths and weaknesses in that domain so that accurate pedagogical decisions can be made from them. The more accurate the student model, the better the pedagogical decisions could be. It should be noted that it is not useful to model everything about the student; just what the ITS can use to teach [154]. The modeller maintains the student model by assessing the quality and correctness of the student's

interactions with the system.

There are two types of student model: the long-term model and the short-term model. The long-term model contains the representation of all the student's interactions with the system and therefore provides an estimation of their mastery of domain concepts. The long-term model can be used to make larger pedagogical decisions, such as choosing the next best concept or problem the student should tackle. The short-term model contains only the representations of the most recent interactions of the student with the system. This short-term model is used to give specific feedback on what the student is doing at that present time.

Quite often student modelling representations are similar to domain modelling representations as with student modelling one is trying to ascertain the student's knowledge in terms of the domain. There are many student modelling representations. Two important modelling representations that have been used with success are MT and CBM. The MT tutors are built on an established theory called Anderson's ACT-R theory [11]. In MT tutors, the domain model is runnable and tracks all the paths the student could take to a correct or incorrect solution. CBM tutors are built on a theory by Ohlsson that examines current solution states and judges whether these states are correct or incorrect [136].

There are also many techniques for student modelling; we point the avid reader to Greer and McCalla [68] for more information. Overlay models overlay the student's knowledge over an expert's knowledge (i.e. as a subset of the expert's knowledge). Differential models are a modification of the overlay model. It divides the learner's knowledge into two categories: what they should know and what they could not be expected to know. This means that all gaps in the learner's knowledge are not equally undesirable. Instead some gaps are what the learner should know, whereas other gaps are what the learner could not know at this stage. A genetic graph is an extension of the overlay model where the model is saved as a type of semantic network. The learner's knowledge is described as nodes in the graph while their learning behaviour is shown by the edges. A perturbation model, unlike an overlay model (where the learner's knowledge is seen as a subset of the expert's knowledge), realises and models the learner's knowledge and beliefs beyond

the range of the expert model (i.e. it models what the student believes about the domain that the expert does not). A bug library can be created of all the incorrect knowledge students have and learners can be modelled against that. As the learner progresses, the perturbation model can be updated according to the bugs in the bug library. Knowledge Tracing [50] is another modelling technique. With knowledge tracing, the tutor maintains an estimate of the probability that a rule has been learnt by a student. Using this probability, the tutor can also tell which rules have been "mastered" and which rules require more practice. There are also other modelling techniques that for example, use fuzzy diagnostic student models. Here statistical procedures are used to show how much a student actually knows, ranging from "no knowledge" to "fully developed knowledge". This paragraph is not an exhaustive list of student modelling techniques; one would be premature in making such a list as variations and new methods are continually being developed.

When a student starts to use the tutoring system for the first time, their model could be empty. It could take some time for the model to show the student's actual knowledge. In the meantime, the ITS might not be performing at its optimum. However, with stereotype modelling, the student is classified into a level of mastery using a pre-test or some test of prior knowledge. This test might be as simple as asking the student to rate their expertise as one of novice, competent, or expert. The student's model is then populated initially according to that stereotype and refined over time as the student interacts with the ITS.

2.1.3 The Communications Module

The communications module determines how the students use and interact with the system. It contains the graphical user interface. It also contains knowledge and rules about types of communications with the student. Its main function tries to answer the question "How should material be presented to the student?". For example, it provides the student with the problem (if it is a problem solving scenario) and a solution workspace. Ideally, the interface will contain all the necessary tools required for the student to solve the problem. It is also important that the interface is intuitive and easily lets

the student understand the goal and context of the current situation. This helps to reduce the unnecessary working memory load on the student who is trying to understand the interface and do peripheral tasks, rather than spending it on the learning activity.

2.1.4 The Pedagogical Module

The pedagogical module is responsible for all the teaching decisions within the ITS. It uses information from other modules (such as the domain module and the student model) to determine what action should be taken next. These actions range in nature depending on what pedagogical objectives the author wants to achieve. This module is responsible for low-level actions (what should the system do right now?) to high-level actions (what should the student be given next?). The pedagogical module houses the pedagogical strategies; these determine the decisions that affect the student's learning. Most of the strategies are hard-coded into the system and depend on the developer of the system. Due to the difficulty of adding new strategies, most ITSs have only one set strategy for each decision they have to make. For example, there might only be one strategy to decide on what the student receives next; there may be many variables that this depends on (such as the student's current knowledge), but still only one strategy. Even though our research affects nearly all the components stated above, we are most interested in the pedagogical module and its functions for this project.

2.1.5 The Expert Model

The expert model is very similar to the domain knowledge model in that it contains knowledge about the domain. However, there is one significant difference. The expert model describes how an expert would represent this knowledge. In problem solving environments the expert model is often runnable, i.e. one that is capable of solving problems in the domain [48, 115, 10] and is thus sometimes referred to as the *problem solver*. The system can check differences between the student's solution and the expert model and give feedback when they differ from each other.

2.2 Tutors Used in this Project

2.2.1 SQL-Tutor

SQL-Tutor [116] is a mature constraint-based ITS, which gives students opportunities to practice relational database queries in SQL (Structured Query Language). It is the first ITS created by the Intelligent Computer Tutoring Group and is used by many students and in courses around the world today. It has undergone numerous evaluations that show high student learning gains [123, 120].

The typical method of teaching SQL is usually in lectures and labs with the students able to practice their skills on a particular Database Management System (DBMS), such as Oracle. This makes the task very complex, as students have to learn the correct SQL concepts, know and understand the structure of the database, and be familiar with the DBMS. Furthermore, most DBMSs give cryptic error messages limited to just the syntax of SQL.

SQL-Tutor is not a DBMS. It is built as a tutor to provide a specific *learning-by-problem-solving* environment for the student. There are just under 300 problems available for the student to solve. Problems are context dependent, i.e. they relate to a particular database. For example, there are many problems available to students from the *Cruises* database. This database relates to ships, the crew, passengers, and the various cruises available. Currently there are thirteen such databases. The student can choose to solve any problem in the system, however, certain problems that are deemed to be within the student's zone of proximal development [178] are suggested to the student. Each problem is assigned a difficulty level by the original expert author. Difficulty ranges from 1 to 9, such that there are non-trivial differences in difficulty between each level.

The student interacts with the tutoring system via the SQL-Tutor graphical interface. Fig. 2.3 shows the SQL-Tutor interface. The SQL-Tutor interface consists of the problem text, a solution workspace, the feedback panel, and the database schema. The problem text is written in plain English and describes the problem. The solution workspace is the area in which the student can create their query solution to the problem at hand. The student is able to create this query in any particular order. In some cases,

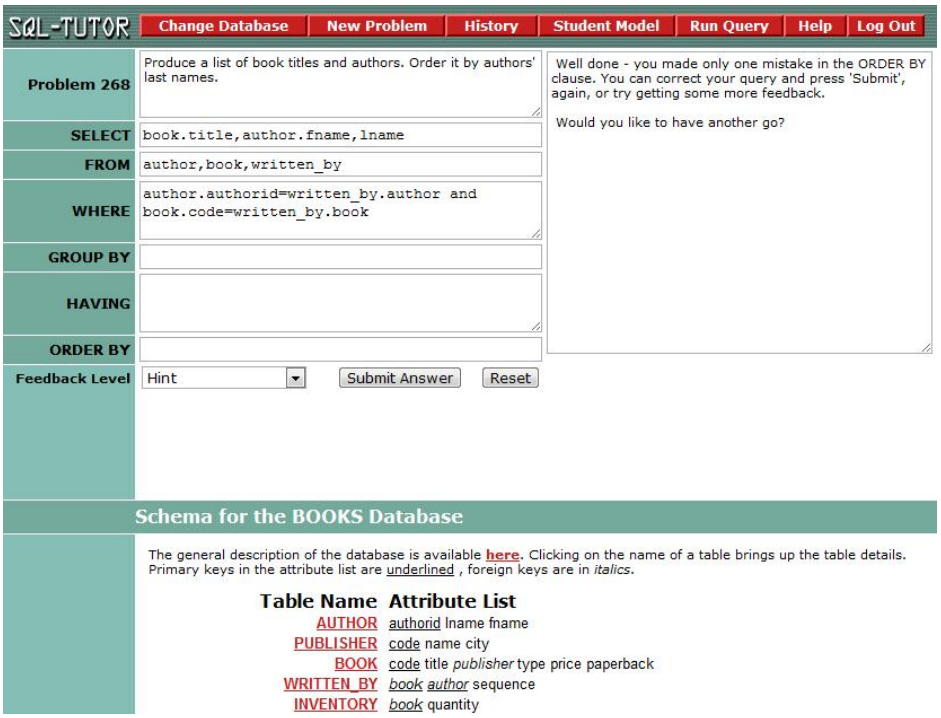

Figure 2.3: SQL-Tutor interface

there might be several different solutions to a single problem. This is easily handled by the constraint-based evaluator. This means that students could come up with novel solutions or equivalent solutions to problems and be assured of their correctness.

A student can submit their solution (or a part of their solution) at any time using the 'submit answer' button. The student is then given feedback specific to their solution in terms of domain knowledge. There are six levels of feedback that give increasingly more help to solve the problem: 1) simple feedback, 2) error flag, 3) hint, 4) partial solution, 5) list all errors, and 6) full solution. This feedback is in plain English and concentrates on syntactic and semantic errors the student might have made in their solution. It provides not only problem-specific help (what is wrong in this solution?) but also domain-specific help (i.e. help at a conceptual level). On each incorrect submission, feedback increases automatically up to the third level. After this, or at any stage, the student can select higher levels of feedback by using a feedback combo box.

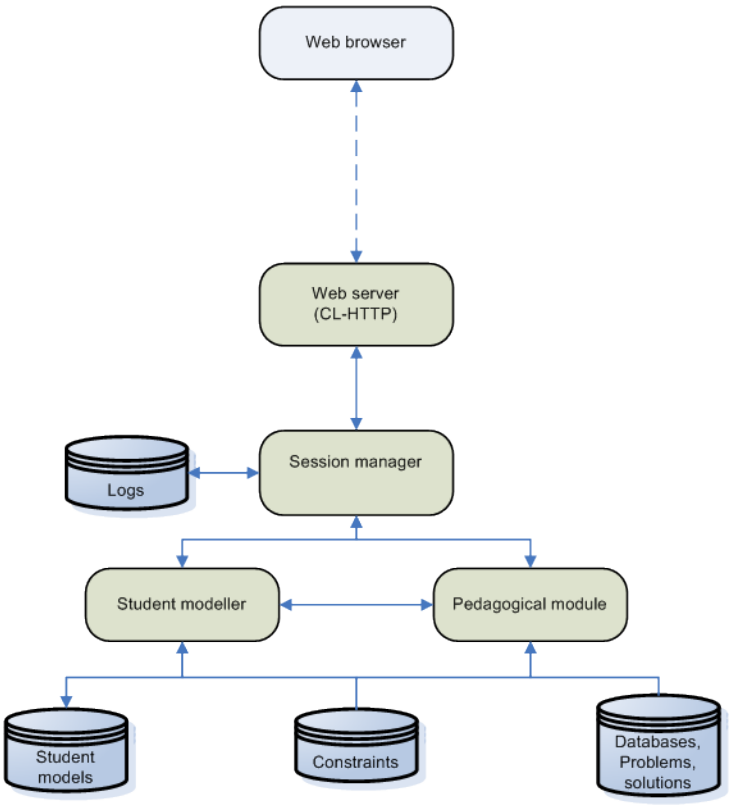

Figure 2.4: SQL-Tutor Architecture, adapted from [116]

The database schema panel is found at the bottom of the window. It shows the current database with all the relevant tables. Students can click on the tables to see the individual attributes. This window helps minimise cognitive memory load of the student, allowing them to concentrate on learning the concept rather than trying to remember trivial aspects of the database.

SQL-Tutor is also connected to live databases with real data. The student can, using their solution, query the database and view the outcome.

SQL-Tutor Architecture

The SQL-Tutor architecture is shown in Fig. 2.4. Students access SQL-Tutor via the internet using their web browser. SQL-Tutor is served on a web server using the Allegro Common Lisp web server. A session manager records each session; a session is described as one student's interaction with the server from the time they logged in till they have logged out or have

automatically been logged out by SQL-Tutor. The session manager keeps logs of all events that have occurred on the server. The constraints contain the domain knowledge base for the domain, SQL. Constraints are explained further in Chapter 5. The student modeller evaluates the student's solution and keeps track of their progress in the student model. The pedagogical module contains the logic to use the problems and solutions (e.g. displaying the correct problem) and the databases.

We used SQL-Tutor almost throughout the whole project as a base to our framework. There is a short section in Chapter 9 where two other tutors, EER-Tutor and ERM-Tutor, are used. We thus briefly discuss them here.

2.2.2 EER-Tutor

EER-Tutor [190, 123, 120] is another mature constraint-based ITS within the family of database ITSs developed by ICTG. It provides the student with many opportunities for learning by practising their Enhanced Entity Relationship (EER) modelling. Similar to SQL-Tutor, it is served on the internet and students can use it via their web browser.

The graphical user interface of EER-Tutor is shown in Fig. 2.5. As with SQL-Tutor, the student is presented with a problem in textual format. From this description of the problem, students are asked to design an EER diagram. This is a very complex task as there are many paths for the student to take to correct solutions. Furthermore, they can start at any point.

The solution workspace is an area where the student can build their EER diagram. The toolbar underneath the problem text contains buttons that students can use to create their diagram. The buttons represent the components of the EER model. The solution workspace is a Java applet, allowing for easy drawing.

Part of the difficulty of creating these diagrams is mentally building the requirements from a real-world scenario description of the problem. To help with this, students double-click on certain key words in the description of the problem to use in their solution. This shows the tutor that the students have carried out the requirements analysis correctly.

At any time during the solution building process, the student can choose

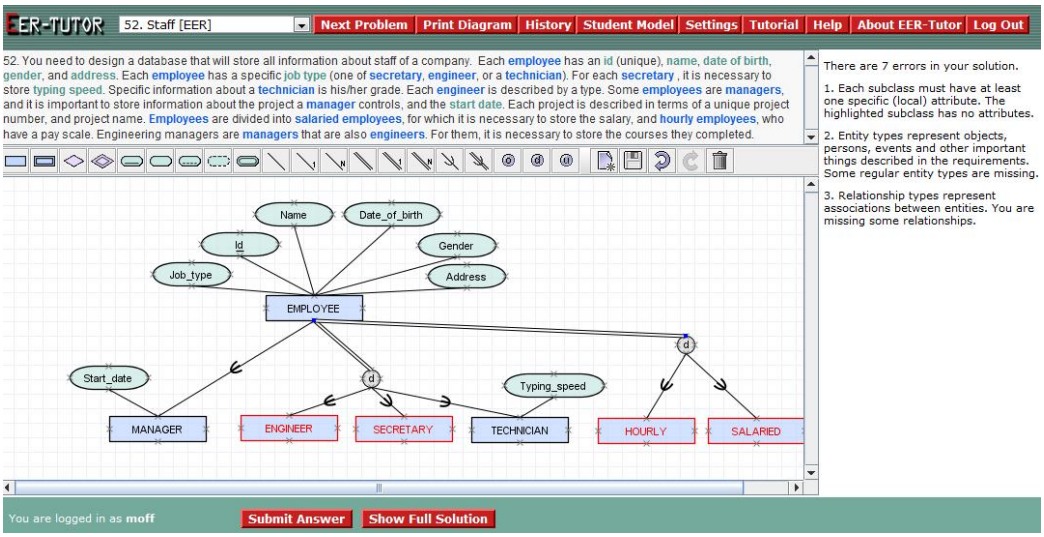

Figure 2.5: EER-Tutor interface

to submit their solution. This gives them feedback (both problem and domain) in the feedback panel. In the main version of EER-Tutor, students can choose the amount of feedback they require by manipulating a combo box similar to SQL-Tutor. In certain versions (such as the one shown), feedback is automatically incremented to the correct level.

EER-Tutor was originally created as a desktop application called KER-MIT (Knowledge-based Entity Relationship Modelling Intelligent Tutor) and more information about the system can be found in [165].

2.2.3 ERM-Tutor

ERM-Tutor (Entity to Relational Mapping Tutor) [112] is another ITS in the database ITS suite used at the University of Canterbury. It teaches students the Entity to Relational Mapping Algorithm. This gives students practice at the seven-step mapping algorithm, which maps the Entity-Relationship diagrams into relational schemas. The seven steps include mapping: 1) regular entities, 2) weak entities, 3) 1:1 binary relationships, 4) 1:N binary relationships, 5) M:N relationships, 6) multivalued attributes, and 7) n-ary relationships. This tutor is quite different to SQL- and EER-Tutor in that it teaches a procedural skill. However, although this algorithm is well-defined

and short, students find this difficult to learn and apply consistently.

The ERM-Tutor is also a web-based tutor. Similar to the other tutors, the main components of the system are the pedagogical module, student modeller, session manager, and user interface. However, ERM-Tutor also has a problem solver that can solve problems that are given to it.

The graphical user interface for ERM-Tutor takes the student through each of the steps separately and sequentially. This allows them to learn the order of the steps and concentrate on getting each step correct before moving on. One part of step 1 is shown in Fig. 2.6; the feedback pane to the right has been cropped to allow for a bigger, readable image.

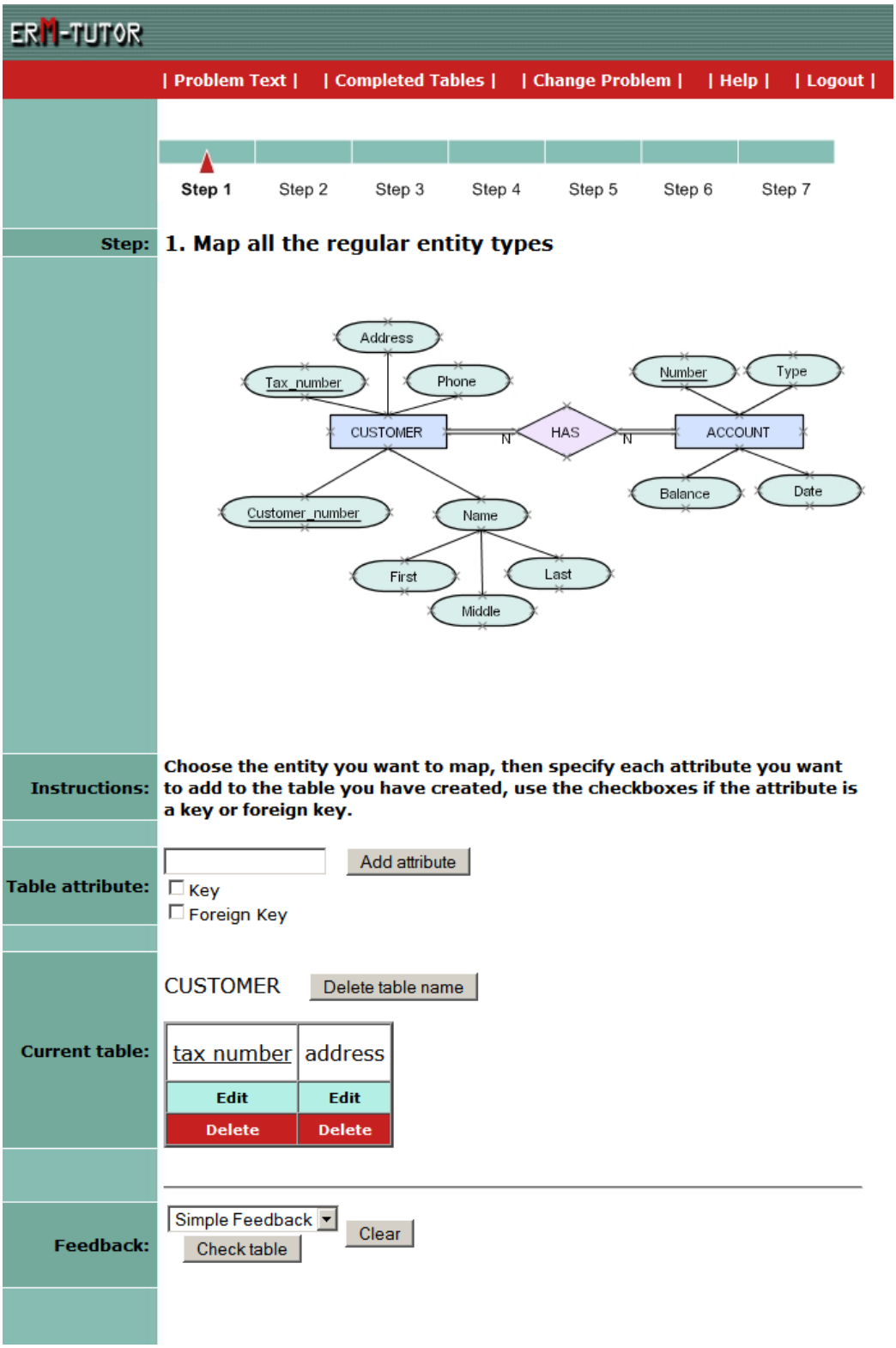

Figure 2.6: ERM-Tutor interface

Chapter III

Pedagogical Background

In Chapter 2, we looked at the background information regarding Intelligent Tutoring Systems. In this Chapter, we focus our attention on background information to do with 1) pedagogical theories, 2) pedagogical strategies, and 3) the attempted use of multiple strategies within ITSs until now. In the Sections regarding the theories and strategies, we do not give a comprehensive and complete listing of all theories and strategies, rather we give the reader a few examples of what we mean by the theories and strategies. This should then better help the reader understand how it fits within the MAPS Framework.

When looking at theories, we browse through Bloom's Taxonomy, Experiential Learning Theory, Social Learning Theory, Communities of Practice, ACT-R Theory, and Learning from Performance Errors. For strategies, we look at Preventing Harmful Gaming of the System, Instant versus Delayed Feedback, Curriculum Sequencing, Open Student Modelling, Using Worked Examples, Tutorial Dialogues, Learning by Teaching, and Problem Posing.

3.1 Pedagogical Theories

Over the years, educators and psychologists have come up with several theories in an effort to comprehend not only how we learn (and therefore how we should teach), but also about how the mind works. These theories were initially categorised using their main differences into Behaviourism, Cognitivism, Constructivism, Design-Based, Humanism, and Social Theories [74]. However, nowadays with the multitude of theories proposed, most theories are difficult to place squarely into one of these categories; they use and build on other learning theories and models.

Pedagogical theories are two-sided: learner-centric and teacher-centric. Some of these theories promote one or either side while others are more based on the workings of the mind during the learning process. Sometimes, these theories are referred to as Learning Theories or at other times as Teaching Theories; we classify all under the term Pedagogical Theories.

In this Chapter we describe a few of these theories. We do this with the reason to show the differences between the pedagogical theories and pedagogical strategies and how we require both, and that one informs the other. We do not necessarily subscribe to any pedagogical theory for this project. In fact, the main reason for this Chapter is to show that there are many theories, and the pedagogical strategies that arise from whatever theory one might use could be used within our MAPS Framework.

3.1.1 Bloom's Taxonomy

Bloom's Taxonomy, or more formally, the Taxonomy of Educational Objectives [35, 153] is a framework that was created to categorise exam questions. Benjamin S. Bloom with the help of colleagues created these categories so that they could share questions, placing them into banks of similar questions in the hope of reducing the load of writing examinations. After a series of conferences from 1949 to 1953 dealing with design of curriculum and testing, it was decided that a taxonomy would be created for three main aspects or domains: Cognitive, Affective, and Psychomotor. The first book to be published regarding the Cognitive domain was published in 1956 under the title: "Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain (Bloom, Engelhart, Furst, Hill, & Krathwohl, 1956)"[35]. This handbook contains the original taxonomy. Even though this taxonomy is used primarily for categorising questions, it is used widely to inform curriculum formation and teaching. The second volume "Handbook II: Affective Domain"[97] was published in 1964. The third handbook was never published. A revised version with changes was introduced to Bloom's Cognitive Domain [164]. We will be referring to the revised version in this project. This Taxonomy is still widely used. We consider it as an example of a teaching theory in this project and look briefly at

the first two domains: Cognitive and Affective. The taxonomy is hierarchical and assumes that the prerequisite concept must be met before moving onto the next concept in the hierarchy.

The Cognitive Domain

The revised version of Bloom's taxonomy has two dimensions.

The First Dimension

In the first dimension, the taxonomy terms are: remember, understand, apply, analyse, evaluate, and create. Each term builds on the previous one, i.e. they were defined in such a manner that the objectives in one class are likely to make use of and be built on the behaviors found in the preceding class in this list.

The first class in the list is *remember*. The types of questions in this class test students on their memory of knowledge/facts. Here the student is tested on knowledge of specifics (e.g. methodologies used), terminology, and universals or abstractions in the field. The second class in the list is *understand*. Here, the student has to demonstrate the understanding of the facts by organising, comparing, interpreting, translating, and explaining them. The next class in the list expects the student to be able to *apply* what they have learnt and understood. They then should be able to *analyse* by breaking up the information into parts and understanding the elements, relationships, and organisational principles that make up the facts. After the process of being able to break up and understand the individual pieces of information, they should be able to judge or *evaluate* the validity of information in terms of internal or external criteria. Finally, they should be able to use this information and *create* new information based on what they have learnt.

The Second Dimension

The second dimension talks about types of knowledge. There are four different types of knowledge: factual, conceptual, procedural, and metacognitive.

Factual knowledge refers to the basic elements of the domain that students should know. This includes terminology and specific details.

With *Conceptual knowledge*, students should be aware of different cate-

Table 3.1: The cognitive process dimension

The knowledge dimension	1. Remember	2. Understand	3. Apply	4. Analyse	5. Evaluate	6. Create
A. Factual knowledge						
B. Conceptual knowledge						
C. Procedural knowledge						
D. Metacognitive knowledge						

gories, classifications, generalisations, and structures within their domain.

Procedural knowledge focuses on the skillset required in this domain. Students should be familiar with algorithms, methodologies, procedures and when to use them.

To have metacognition is to be self-aware and objective about what one is doing in the domain and how they are doing it. *Metacognitive knowledge* focuses on strategic knowledge, understanding contextual and conditional knowledge and self-knowledge.

The Cognitive Process Dimension

The revised version of Bloom's taxonomy (the cognitive process dimension) is therefore a table (2 dimensional) with the classes as stated above (see Table 3.1). As examples, if a student was asked to remember a fact about the domain, this would be categorised in cell A1. If they were asked to explain the consequences of one principle's effect on another, this would be categorised in B2, as they not only have to be able to explain (which comes from understanding), but they would also have to understand conceptual domain knowledge and not just the facts.

Changes to the original version

There were a number of changes between the original version and the revised version [95, 96]. Firstly, the original taxonomy was named as nouns i.e. knowledge, comprehension, application, analysis, synthesis, and evaluation, whereas a major change in the revised version was to change these to verbs: remember, understand, apply, analyse, evaluate, and create. This was done to emphasise that each of these elements had to be done by the student. Secondly, the last two items were switched (evaluate now comes before

create). Thirdly, *synthesis* was changed to *create* to emphasise that it does not only mean synthesising and putting together the knowledge gained but also creating something quite new and novel. Finally, the one dimensional taxonomy was made into a two dimensional table by adding four different types of knowledge on the other axis: Factual, conceptual, procedural, and metacognitive.

The Affective Domain

Affect describes how people react to things emotionally. It is feelings oriented. This domain looks at how students grow in emotions, attitudes, and feelings. It has five classes: receiving, responding, valuing, organising of values, and characterisation by value. As with the previous domain, objectives of a class build on preceeding class's behaviours [58].

In *receiving*, the student is passive and simply pays attention. Learning is minimal. Once the student starts to react to the stimulus and interact with it, they are in the *responding* class. The student is actively participating in the learning process in this state. During *valuing*, the student links some value to what they have learnt. Pieces of information now have different amounts of value. The student then takes the different pieces of information (including their value) and using already known information, *organises* it within their memory schema. When *characterising* occurs, the student associates some belief with this information so that it now exerts influence on his/her behaviour. This means it has now become a characteristic.

In both domains (the cognitive and affective), the teacher wants the students to move up the dimensions. The process of keeping interest alive and thus proceeding to the next stage is referred to as motivation.

3.1.2 Experiential Learning Theory (ELT)

In the early 1980s the theory of Experiential Learning started to gain prominence with the work of Mezirow, Freire, Kolb, and Gregorc. They stressed that at the heart of all learning lies our critical reflection of experience [145]. Instead of learning being an external process (where the teacher pushes knowledge onto the students), this theory states that learning comes from

within the student. The most influential in this theory is David Kolb, whose book *Experiential Learning* [93] is the best known presentation of this approach. He states that the original founders of this approach were John Dewey, Kurt Lewin, and Jean Piaget. This theory has been the underlying model used in developing adult education theory [79, 183] and organisational learning [57]. Amidst criticism of some of its basic principles (e.g. [111]), it is still widely used today.

In this theory, four stages form a learning cycle, formally known as the *four-stage model of experiential learning* (see Fig. 3.1). Students start with some Concrete Experience (CE) that affects their *feeling*; i.e. they experience it fully. This leads them to think about and reflect on their experience, moving them to the second stage: Reflecting Observation (RO). Here students are involved in *watching* and reflecting on what they have experienced. The students then put some thought into what they have experienced and reflected upon to come up with an Abstract Conceptualisation (AC); they do this by *thinking*. Finally, they try something new using their new abstracted knowledge to see if what they experience is what they expect. This is the fourth stage: Active Experimentation (AC), which they get to by *doing* or performing some actions. The results of this experimental stage guide the next stage to experience something new; another concrete experience.

With this model Kolb describes two continua between dialectically opposed modes of adaptation to the world; learning occurs when these conflicts are resolved [92]. The first is between Active Experimentation and Reflective Observation and is called the Processing Continuum. This is the way we *do* things; i.e. we watch how things are done then try it for ourselves using the model we have formed. The second continuum is between Concrete Experience and Abstract Conceptualisation and is called the Perception Continuum. This is the way we *think* about things. The feelings generated by the experience makes us think about an abstract model that is creating this experience.

In experiential learning, the student is the centre of the learning process. The teacher is merely a facilitator. The student finds the experiences and goes on a journey of experiencing, forming abstract models, and experimenting to see if what they have formulated is correct. This is where the learning takes

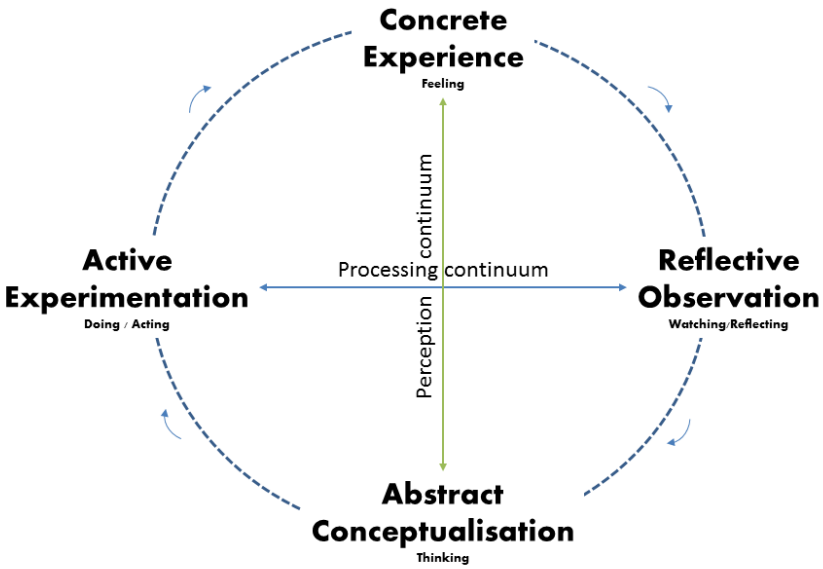

Figure 3.1: Kolb's Learning Cycle

place; inside the student. The student is continually moving from *experience* to *critical reflection* to *abstraction* to *active experimentation* [90].

Learning Styles

Even though the cycle is the idealised model and learners "touch bases" as they proceed through their learning, each individual has their own learning style. This describes the position on the cycle where learners are generally most comfortable. This learning style depends on several factors, such as hereditary factors, life experiences, and the demands of our present life [92]. The Learning Style Index (LSI) was created to find out a learner's learning style [94].

In the initial work of the LSI, a learner could be in one of four learning styles: assimilating, converging, accommodating, and diverging. The learning styles are between the main stages on the learning cycle. People who found themselves most comfortable between CE and RO had a diverging style which consisted of primarily feeling and watching. Those who were

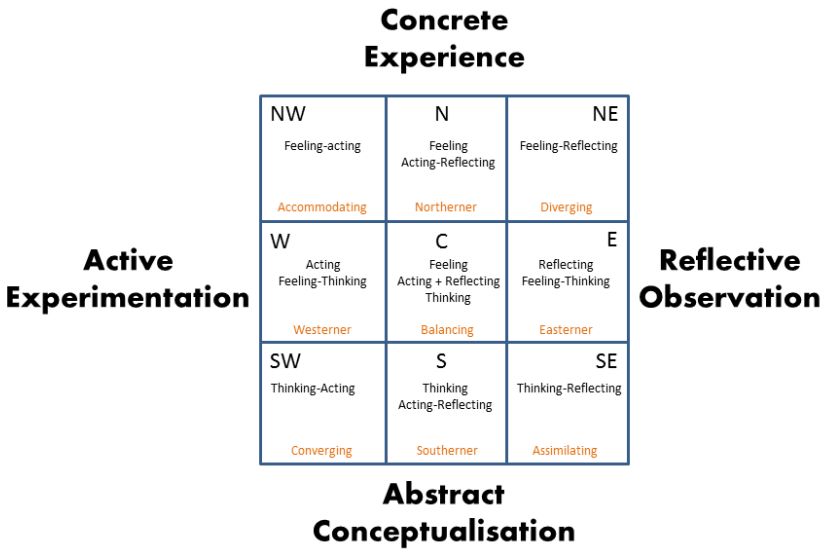

Figure 3.2: The nine-region learning style type grid [92]

between RO and AC had an assimilating style, prefering to think and watch. The converging style is between AC and AE, where people prefer to think and do. Finally, the accommodating style is between the AE and CE; these people prefer to feel and do.

More recent research by David Hunt and colleagues showed that there are in fact nine learning styles [75](see Fig. 3.2). The extra learning styles are named using compass headings, e.g. the four extra states they found were Northerner, Southerner, Easterner, and Westerner. The Northerner emphasises on feeling (CE) while balancing acting (AE) and reflecting (RO). The Southerner emphasises thinking (AC) while balancing acting (AE) and reflecting (RO). The Easterner emphasises reflecting (RO) while balancing feeling (CE) and thinking (AC). The Westerner emphasises acting (AE) while balancing feeling (CE) and thinking (AC). A further style, called a balancing style, was identified in 2002 which integrates AC, CE, AE and RO.

Each learning style has a particular learning attitude associated with it. For example:

> "The learning strengths of [the Northerner] style are a capacity for deep involvement while being comfortable in the outer world of action and the inner world of reflection. This person has difficulty in conceptualising or making meaning of experience; consequently, the cycle runs from feelings to reflection (which remains unconsolidated) to action. The consequence of this Northerly pattern is that the flow is discontinuous and the actions are poorly organised since they are not informed by the foundation of the AC meaning."[92]

Even though the ELT focuses on the student, it implies a certain teaching theory. This is further accentuated when thinking in terms of learning styles. For example, "How would a teacher (faciliator) guide someone from an assimilating style?" or "How much (and type of) help would a teacher give a Northerner?"

3.1.3 Social Learning Theory

Social Learning Theory (also known as Social Cognitive Theory) proposes that people learn from one another, socially, by observing, imitating, and modelling one another. Albert Bandura posits that people learn from observations of others' behaviour and the outcomes of that behaviour. In future, the observers then model that behaviour (or avoid it), using what they saw and learnt as a guide [26]. Feedback from one's own performance when applying this behaviour leads to self-corrective adjustments to refine the behaviour even further [25].

There are four necessary conditions for modelling to be effective [24]:

1. **Attention**: The more attention paid, the better the person will be able to model the behaviour. However, many variables affect attention, such as past reinforcement, current arousal level, etc.

2. **Retention**: This has to do with memory. The more one remembers what they paid attention to, the better the modelling will be. Through coding and symbolic rehearsal, experiences are transformed to become enduring performance guides.

3. **Reproduction**: Being able to reproduce what the person paid attention to, increases the ability to model the behaviour. The integration of actions derive new response patterns.

4. **Motivation**: The student needs to have a good reason to model the seen behaviour. The greater the motivation, the greater the modelling behaviour.

This theory is actively judgmental and constructive (i.e. it actively uses cognitive processes), rather than just being a mechanical copying process [24]. Bandura believes that personality is an interaction between three components: one's psychological processes, their behaviour, and the environment. For instance, people's actions construct the way their environment is fashioned. People's actions are a result of the psychological processes that are cognitively formed in their minds. The way the environment is formed also plays a role on how people view it (and the actions that caused it to be in that state). As such, one event can be a stimulus, a response, or an environmental reinforcer depending on the place where the analysis begins.

In contrast to the behaviourist models that preceeded this theory, Bandura is a great believer in self-efficacy. Self-Efficacy beliefs promote the desired changes in one's life (environment and actions). Efficacy beliefs determine 1) the goals people set for themselves, 2) the amount of effort they put into the endeavour, 3) their staying power amidst adversities, and 4) how formidable they perceive their impediments to be [27]. He uses the example of overcoming substance abuse. The overcoming of drug and alcohol abuse is correlated to a person's self-efficacy. Self-efficacy was closely related to collective agency, where the user was able to break ties with the previous drug-induced social networks, join new social networks, and used cognitive processes effectively to extinguish the cravings [27].

Bandura pointed out four sources of efficacy information: performance accomplishments, vicarious experiences, verbal persuasion, and emotional arousal [25]. The greatest amount of learning comes from participant modelling, where the student is actually involved in the experience to model the behaviour. This leads to greater performance accomplishments, which leads to greater self-efficacy. Vicarious experiences were the second best form of

learning, where the learner simply watched a live model behave in a particular manner and the outcomes of that behaviour. Verbal persuasion was through suggestion, exhortation, self-instruction or interpretive treatments. This, on its own was not as successful as the above two, but could be used in conjunction with the other sources of efficacy. Finally, the more emotionally aroused, the less self-efficacy is witnessed. Attributing blame on external factors, relaxation, symbolic desensitisation, and symbolic exposure are the modes of induction for emotional arousal.

He used this in experiements with people who had severe phobias of dogs [28] and snakes [25]. For example, in the experiment with children (3-5 years old) who had a severe phobia of dogs, he showed that the phobia could be extinguished by learning vicariously through a peer. He had four conditions (2 treatment and 2 control). In the first treatment group, a dog was brought into the room with the kids, who were in a positive context (having a party). The dog was taken through certain tasks by the model, a 4 year old boy. The second group was almost identical to the first, except that the context was neutral (i.e. the kids were simply watching and not having a party). In the first two conditions, the fear-provoking properties of the modelling displays (tasks performed by the model) were gradually increased from session to session. The third condition was in a positive context with the dog present, but the model absent (i.e. no modelling behaviour was performed). The fourth condition was simply a positive context (the party) without the dog or the model. The phobia was measured in each case before and after the experiment. In this experiment, there was a significant difference between the first two treatments and the controls, where the children in the treatment groups experienced the extinguishing of the phobia. There was no significant difference between the treatment conditions or between the control conditions.

Bandura also found that learning in this manner translated not only to learning that particular skill, but it generalised to familiar skills. For example, those whose phobias to one type of snake were extinguished not only were comfortable with that snake, but also with other types of snakes and even more comfortable in social situations when previously they were not. This is what led him to formulate the theory of self-efficacy and behavioural

change.

3.1.4 Communities of Practice

Communities of Practice (CoP) [186] [1] is a form of social learning, in that it involves a collective group of people learning, almost incidentally, from each other. For CoPs to occur, there must be a *domain* of interest that gives the community its identity; a *community* of people who meet fairly regularly to converse and better their understanding of the particular domain; and there must be a *practice* (i.e. it is not just an interest club - the people are practitioners in this domain) [185].

The phrase Communities of Practice was recently coined although it is an old concept. There are a variety of activities by which learning occurs in a CoP. This could be done by problem solving together, requesting information from someone within the community, seeking experience, reusing assets that were already created, getting together and utilising synergy, discussing developments in the field, documenting projects and other things important to the group, visiting each other, and mapping knowledge and identifying gaps [184].

3.1.5 ACT-R Theory

The Adaptive Character of Thought (ACT-R) theory [12] is a theory of the mind that explains how the mind stores and uses knowledge. It divides knowledge into two separate categories: declarative and procedural knowledge. Declarative knowledge is knowledge about facts (in a domain) that a person can remember and report, whereas procedural knowledge is knowledge on how to use these facts to perform skills (such as solving problems).

Declarative knowledge is represented as small primitive units called chunks that are schema-like structures. Each chunk has an *isa* pointer to its category and a number of other pointers to the contents of the chunk. A chunk of information is shown in Fig. 3.3 [13]. This particular chunk is an addition fact that stores "three plus four equals seven". This can be textually written as:

[1] More info: http://wenger-trayner.com/Intro-to-CoPs/

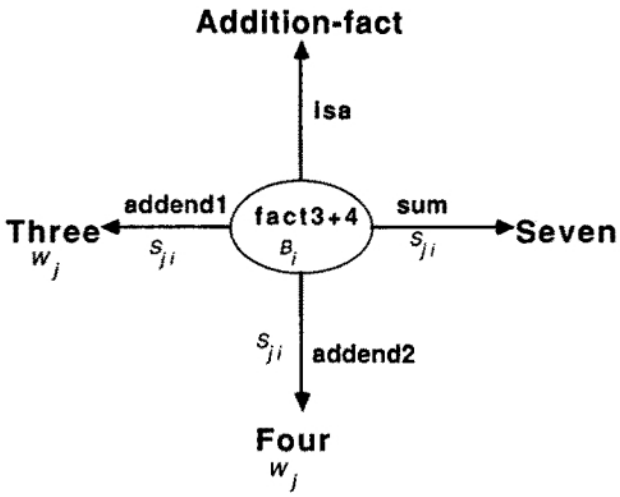

Figure 3.3: Representation of an ACT-R chunk [13]

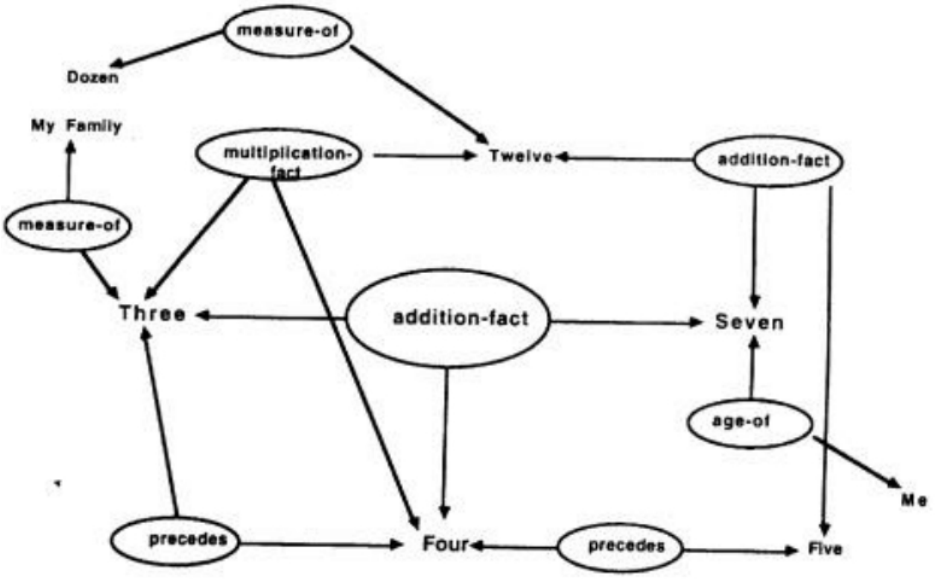

Figure 3.4: Graphical display of a chunk encoding the fact 3 + 4 = 7 [11]

fact3+4

isa addition-fact

addend1 three

addend2 four

sum seven

Chunks of declarative information are not independently placed but are linked to other chunks of information. In Fig. 3.4, the same declarative information is placed within a network of chunks. Chunks primarily originate from the environment. This means that knowledge structures result from environmental encodings [13]. As ACT-R moves its attention over the visual array, objects fall into the *spotlight* and are recognised as objects. These objects are then synthesised and ready to be used as chunks. As another example [13, 14], a potential chunk encoding the letter H could be:

Object

isa H

left-vertical bar1

right-vertical bar2

horizontal bar3

Procedural knowledge is represented by rule-like units called production rules. These are *condition-action* units which respond to various problem solving situations with specific cognitive actions. The condition of the production rule specifies when a rule should execute (fire).

As an example [11], if a student was at the point below:

$$\begin{array}{r} 534 \\ +248 \\ \hline 2 \end{array}$$

then, focusing on the tens column, the following rule might apply:

IF the goal is to add n1 and n2 in a column
and n1 + $n2 = n3$
THEN set as a subgoal to write n3 in that column.

This production rule will then work on a declarative chunk, such as the one we encountered before ($3 + 4 = 7$) in Fig. 3.3. At this stage, the production rule could have a subgoal to take care of other things like processing a carry.

ACT-R is both a theory of learning by doing as well as a theory of learning by example. Production rules are learnt by *analogy*. For analogy to work, first there needs to be a situation where this production rule is required; second, there has to be some example of the solution of such a goal.

The activation of a chunk depends on how well it was learnt and how close it is to the current context the student is facing. Activation is dependent on *base-level activation* (which is the amount of previous learning) and *associative activation* (the closeness to the current situation). The base level activation is increased if the chunk is encountered more recently and more frequently. The associative activation is dependent on the strengths of associations of the individual chunks and how they are contextually primed for this current situation.

3.1.6 Learning from Performance Errors

Learning from Performance Errors is a pedagogical theory by Ohlsson, primarily to do with skill acquisition during unsupervised practice [137]. This theory proposes that when a student performs an action and makes an error, it sets the scene for learning to occur. Errors are caused by over generalised procedural knowledge structures. Even though the student might have sufficient declarative knowledge, they might still make a mistake as they have to learn how to apply the declarative knowledge.

For learning to occur in this situation, one has to *catch* the mistake, and then *correct* it. To catch the mistake, either the student has to have sufficient declarative knowledge to realise that the actual outcome was different to the expected outcome or a third party (such as a teacher, ITS, etc.) has to make

the student aware. Often environmental cues make the student realise that something has gone wrong. To correct the mistake, the student needs to specialise the overly general knowledge and understand how to apply it.

When performing skills, ideally a person keeps finding themselves in particular *situations*, from which an *action* would take them closer to their *goal* state. They keep performing the relevant actions in various situations to end up in the goal state. Each of these linked elements of situation, action, and goal are called a production rule. More formally, a production rule is such that, for any particular goal (G), and a particular situation (S), there is an action (A) that would take you to the goal state.

$$G, S \rightarrow A$$

This causes an action to occur to take the person to the next situation. All these situations from the initial state to the final state can be mapped out and form a *situation tree*. From certain situations, there might be different plausible actions that one could take. At each point in the situation tree, a person finds themselves *perceiving* their situation and the rules that might need to be applied, making a *decision* about what needs to be done next (which rule to use), and then *acting* on it. This is called the perceive-decide-act cycle.

There are two views of errors: the objective view and the subjective view. The objective view comes from having a global perspective of the situation tree and knowing when a student has taken the wrong path (wrong action) from a particular position in the tree. This is not always useful to the student as the student does not necessarily have the situation tree in their minds. The subjective view of the error is the view from the student's perspective. This is when they recognise that something that ought to be true is not; i.e. what they expected is different to what they have. In some cases, this is quite prominently realised, particularly if there are environmental cues (such as burning food or road signs); in other cases, it is more subtle (say, in programming, or in statistics). A subjective error is not always an objective error as a student might perceive a correct state to be incorrect (i.e. they have a misconception). Similarly, an objective error is not always a subjective error, as the student might accept errors as being correct.

In this theory, declarative knowledge is seen as being evaluative (judgemental). It does not describe or explain what should be the case, but it helps the student judge the correctness of their solution. The fact that students can catch themselves making a mistake means that there are two distinct processes going on in parallel: the action and the judgement of the action. This is called the dissociation hypothesis. Because the knowledge used for each process is different, students can know they have made a mistake either during or after they have completed the action.

In the evaluative process, knowledge is stored in such a manner (as constraints) that it can be used to judge or evaluate the student's current state. A *constraint* is said to be a fact or principle that is true in the domain, that acts as a means to a judgement, i.e. it judges whether the student's solution is correct or not. A constraint is the fundamental element of declarative knowledge. If the constraint is relevant for the situation the student is in, then as long as the constraint is satisfied, the student is on a correct path; the violation of constraints means that the student has made errors [138].

When a constraint has been violated, it means that the underlying knowledge used to make the particular action is wrong. This theory states that it is because of overly general knowledge. When a student starts a skill, they are generally problem solving. When problem solving, if there is no production rule that is close enough to their goal/situation, then they must generalise and try a more general production rule. Production rules are modular and therefore can be altered individually. As a student tries these rules and makes errors, s/he specialises the rule base until a correct action is found. Finding the correct action might be through trial and error or by increasing their declarative knowledge. They now have a better suited, more specialised set of production rules; i.e. they have learnt how to solve this particular part of the problem by detecting and correcting their own error. It could be that in generalising and specialising their knowledge, students may give up or try random things leading them to fail. In these cases, teaching strategies must be able to deal with such cases and minimise their occurrence.

There are three sub-processes to correcting one's error. 1) Part of correcting one's error is to detect exactly what went wrong. The error signals do not always immediately follow the error; i.e. it might be some time after making

the error that the student has realised that they have made the error. An example of this is navigating a vehicle. By the time the driver has realised that they are on the wrong route, they might have made a mistake quite some time before the error was noticed. Being able to find the erroneous rule is called *blame assignment*. 2) *Error attribution* is the process of finding the situational factors that played a part in creating this error. 3) After blame assignment and error attribution has been done, the rule can be revised and corrected. A rule is specialised repeatedly until it is activated only when appropriate.

The Constraint-Based Modelling (CBM) technique [121, 136] used in Intelligent Tutoring Systems has evolved from this theory. Constraints are atomic domain principles containing a relevance and a satisfaction condition. If the relevance condition is true (i.e. if the constraint is relevant in this situation), then the satisfaction condition must also be true, otherwise the student has made an error. More information on constraints can be found in Chapter 5.

3.2 Pedagogical Strategies

In the previous Section, we discussed a few pedagogical theories. These theories give us some idea on how our mind works when it comes to learning. These theories are broad and overarching and although they might give the teacher some idea on outcomes, they do not provide actual methods to implement them. In this Chapter, we look at some of the tools (pedagogical strategies) that educators have used to achieve the desired learning outcomes. It is these strategies that both human tutors and ITSs require, to teach the way they do. These strategies may comply with one or more of the pedagogical theories.

We have divided strategies into two groups: domain-specific and domain-independent. The domain-specific strategies depend on each domain in which they are taught (e.g. how does one do multi-column subtraction?) and are considered outside the scope of this project. We are therefore only concentrating on the domain-independent strategies. Quite often, strategies are complex, i.e. they are made up of other simpler strategies. In these cases,

the educator has to be aware of all the most basic atomic strategies to enter them into the MAPS Framework.

The reason for this Chapter is to give the reader a flavour of what we mean by pedagogical strategies and the very wide variety of them, so that it makes sense when we refer to our MAPS Framework and the strategies that could be used. It is not a comprehensive overview of all existing strategies. For all strategies mentioned, we also look at current ITS applications.

3.2.1 Preventing Harmful Gaming of the System

Gaming the system is when a student exploits the system's properties rather than using their own knowledge to solve the problem to get ahead in the tutoring coursework within the ITS [20]. One example of gaming the system is when a student repeatedly and in quick succession hits the hint button till it bottoms out and gives them the solution for that step. Gaming the system in this manner leads to poor performance [21]. This type of harmful gaming, primarily of help abuse, has been found in ITSs [130, 29]. It has been found that students use different levels of help, with some students opting for the full solution on every attempt [103]. Students who are disuaded from using help as often do better than those who use help often [130]. To detect when gaming is occuring and stop the process, a number of detectors have been created for the various types of ITSs [23, 19, 29]. To stop harmful gaming, an animated agent was created called "Scooter the Tutor" with the gaming detector on the backend [18]. The idea was to prevent gaming and allow the students who game the opportunity to learn, while changing the tutor minimally for those who do not game. As students game the system, Scooter displays increasing levels of displeasure. If the student gets an answer correct using gaming, Scooter then gives them supplementary exercises to make sure they understand the concept. For students who are not gaming, the system does not change. Using this system, the students engage in less gaming. Students who receive many supplemental exercises also have better learning.

3.2.2 Instant versus Delayed Feedback

A difficult challenge facing ITS developers is when to provide the student with feedback. This is particularly important in problem-solving and other interactive environments. Researchers argue the pros and cons of giving feedback as soon as the student has made an error (instant feedback) versus giving them feedback only when it is asked for or needed (delayed feedback), and whether the timing of feedback makes any difference [175]. Sometimes, instant feedback is given after formative assessment (so that the incorrect steps are not rehearsed) but delayed feedback can be given after summative assessment (to encourage reflection) [140]. As examples, Andes Physics Tutor [176] provides instant feedback on the correctness of each step of the solution. On the other hand, SQL-Tutor [116] waits until the student submits a solution before providing feedback.

Providing instant feedback does not allow the student to stray too far from the solution path, and keeps them close to the route to solving the problem. This makes blame assignment a lot easier as the error and feedback for the error are close in respect to time. This feedback could be good for novices who need closer monitoring. However, with instant feedback, the student could be interrupted each time he/she strayed off the correct solution path. This could stop the student from trying something different or new. Delayed feedback allows the student to think out their solution more fully before submitting it for feedback. This could be better for more experienced students. People do not necessarily learn because two events are temporally located; they learn when they realise there is a correlation between the two events [24] (i.e. feedback does not have to be temporally co-located with the error). Delayed feedback also allows for novel solutions.

3.2.3 Curriculum Sequencing

Curriculum sequencing is the process of finding the next best topic (concept, problem, etc.) for students to learn. Curriculum sequencing can be divided into concept sequencing and task sequencing [37]. In classroom teaching, curriculum sequencing is necessarily linear and is usually prepared even before the course begins. This is because the teacher is teaching many students

at once and cannot take into account one student's deficits or strengths over another student's. Good teachers devote considerable effort to adapting such planned sessions while including opportunities for individual interaction with students and personalised feedback. However, with a one-to-one student-to-teacher ratio, teachers dynamically and individually extend the student's knowledge step-by-step by finding the next best learning task. They do this using their knowledge of the student (e.g. the student's current knowledge in the domain) and their teaching expertise.

In e-learning technologies, curriculum sequencing can be done dynamically, non-linearly, and at an individual level. The complexity of curriculum sequencing in e-learning technologies differs greatly. On one extreme, a small amount of dynamic, individualistic curriculum sequencing can be done in LMSs. For example, Moodle's Lesson Module [2] allows the author to build a curriculum (pages of content), but also to have question pages that add choice to the curriculum. Depending on the student's answer, they are redirected to a separate page that the teacher pre-picks. This means that students do not all follow the same path through the system.

In a more intelligent web system, such as MANIC [163], the next best topic is calculated using a more complex method. In MANIC, links are constructed between connected topics, forming a semantic net. Links represent *prerequisite, corequisite, related* and *remedial* relationships. Marks from quizzes show how well students understood each concept of the topic presented. There are other factors that also inform the student's student model, such as time spent on task, whether both the topic was viewed and heard (audio is also available), etc. Links and topics are weighted according to their importance to the student. All of this helps to figure out if a student has studied the topic and its subtopics well enough. The three scores – quiz performance, study performance, and reviewed topics – are combined into a single value that indicates how well the topic is learned. The next best topic is selected using this student model [162].

SQL-Tutor [116] is a constraint-based ITS that supports task sequencing. The student model is based on constraints (domain facts). SQL-Tutor is

[2] `http://docs.moodle.org/22/en/Lesson_module`

micro-adaptive in that it responds to the behaviour of each individual student and in fine detail. Every submission that a student makes of their solution could have many relevant constraints. Thus the adaptivity is done at the constraint level. The tutor keeps track of the student's progress through the system in such fine detail that it is able to provide "the next best thing" at any time. For example, at the end of a problem, the system can suggest the next best SQL clause and within that clause, the next best problem suited for that particular student. During problem solving, when a student makes errors, the tutoring system can provide hints about what the student could do next to solve the problem.

3.2.4 Open Student Modelling

A student model is the system's knowledge of the student. The ITS uses this knowledge to make decisions, such as pedagogical decisions, about each student. The student model is usually in formats that are not easily readable or comprehendable to humans, as its primary goal is to serve the ITS (not humans) with information. Research into metacognition shows that opening up the student model so that the student can view their own model is beneficial [119, 87, 69]. This has its own challenges. How should a student model be visualised? Are there different types of visualisations that would be beneficial depending on the student's expertise? Current visualisations include skill meters [119], conceptual graphs [55], hierarchical tree structures [87], concept tags, concept hierarchies, complex treemaps, [106] etc. With each visualisation, there is a trade-off between complexity of the visualisation and amount of detail. Does this affect some groups of the intended audience (e.g. should novices be given simpler visualisations than experts?) or do certain groups consult different visualisations for different purposes [65]? Are there times when the system should prompt the student to look at their student model? These are all open-ended research questions about pedagogical strategies based around open student models.

It has been argued that student models should not only be open but also scrutable [89]. The student should be able to look at their model and not only see what the system thinks of their knowledge, but also get an idea

of how it got this point of view [88]. This leads to the idea of negotiable student models [56]. This is where the student, after looking at their model and disagreeing with some aspect of it, negotiates with the ITS about that part of the representation of their knowledge in the model. One way this has been done is by allowing the student the ability to alert the ITS about which part they want to negotiate [170]. The ITS then provides the student with the opportunity to *prove* that they know this knowledge by presenting him/her with a problem or some form of task that relates to that concept. If the student answers it correctly, the student model is updated accordingly.

3.2.5 Using Worked Examples

The use of worked examples is when a student is provided with one or more worked examples during their learning session. A worked example is a step by step demonstration of how the problem is solved by an expert [167].

The use of worked examples was popularised by Sweller and colleagues as they presented the Cognitive Load Theory (CLT) [166, 129]. The CLT states that while learning, there are certain loads placed on our working memory. These loads affect how much and how well we learn. The working memory has a limit, constraining it to processing 7 ± 2 new chunks of information at any given time [113]. There are three different types of cognitive load: intrinsic, extrinsic, and germane [169]. Intrinsic load is the load posed by the problem itself and therefore, cannot be changed. It is related to the inherent difficulty of the problem (e.g. the number of steps required to solve the problem). Extrinsic load is the load posed by everything else in the environment that the student has to deal with; this load can be altered. In computer-based systems, an example of extrinsic load might be the complexity of the graphical user interface. Germane load is the load related to learning the specific concept or doing the necessary task. This is when schemas in the mind are being modified, created, or automated. Similar to extrinsic load, germane load can be manipulated.

Quite often the reason given for using worked examples as a pedagogical strategy is the "worked example effect" [168]. The worked example effect is when students who study worked examples do better than students who only

solve problems. The worked example effect can be explained by the CLT. To do this, we must first study the four principles of the CLT [167, 150] given below.

1. *The information store principle*: Long-term memory plays a large role on cognition. How we do things (think, perceive, solve problems, etc.) is dominated by the contents of our long-term memory.

2. *Borrowing principle*: Realising the importance of the long-term memory, we must consider how information is placed into it. Items are placed into the long-term store by borrowing it from another long-term store. This could be knowledge from other people or from other materials. However, borrowing the items has a random element to it. When we borrow these items and assimilate it into our current knowledge, we try to assign meaning to the items. We also test whether this information and the new meaning is viable.

3. *Randomness as a genesis principle*: The borrowing principle does not create "new" information, as it simply combines two sources of information. However, when a person tries to solve a problem, information is created as the person randomly chooses moves and tests it for effectiveness in that current state. The effective moves may be subsequently incorporated into long-term memory. This principle is only used when the borrowing principle cannot be used. All knowledge in long-term memory has either been borrowed or randomly created.

4. *Narrow limits of change principle*: Because getting items into long-term memory involves a significant amount of randomness, limits must be placed on how much new information can change at a given time. This is done by placing a limit on our working memory for new information [51, 113]. This limit is not valid for information that is already stored in the long-term memory.

Basic problem solving is a prime example of the random genesis principle, whereas providing students with worked examples is an example of the

borrowing principle. The borrowing principle is much more effective and efficient, hence the worked example effect.

There has been much debate about the existence of the worked example effect. For the current state of research in this field, we refer the reader to [108]. We briefly discuss five situations which have been seen in literature to explain why a framework such as ours (where pedagogical strategies can be closely controlled and adapted) is necessary.

1. **Tutored versus non-tutored problem solving**. In the original papers, Sweller and colleagues referred to problem solving without any instruction on pen and paper. Today, with the use of ITSs that adapt instructional feedback at every step and even provide forward hints, problem solving is very different. Students could even bottom out on each step (and thus see the solution) and create their own examples. The amount of randomness in problem solving within ITSs today is much less than originally predicted.

2. **ITSs versus non ITSs**. Much of the research has been conducted without the use of ITSs. This means that the cognitive loads on the student could be very high during problem solving. ITSs however, adapt to each student, guiding them through difficulty levels of problems and provide adaptive, instructive feedback, specialised for that student's current knowledge. This means that both the intrinsic load is at the right level for the student and that the germane load is kept low.

3. **Split attention**. In some studies, students were given more materials to refer to, as they either solved problems or viewed worked examples. Conflicting results were then gathered. The split attention effect [44, 45] says that when learners have to direct their attention to more than one place at a time, their learning will suffer. As an example, Ward and Sweller [179] found that sometimes students had difficulties when the formulae were separated from (were below) their diagrams; an integrated diagram worked best with all the learner's attention devoted to one place. This varies the cognitive load on the student.

4. **Self-explanation**. In some studies, students were asked to self-explain the solution (e.g. after they saw a worked example). Self-explanation enhances learning [46]. However, this produced confusing results. If a student already had a high cognitive load before the self-explanation question and were asked to self-explain above and beyond that, their performance would have dropped as this would have put an even greater load on their cognition.

5. **Redundant information**. It has been found that providing non-novices with examples does not make them more effective; in some cases it could have a negative effect. The non-novices already know the information and the worked examples simply provide redundant information. This is called the expertise reversal effect [82, 83]. Instead, it is better to scaffold the worked examples and provide more problem solving activities as expertise increases [143].

3.2.6 Tutorial Dialogues

Human one-to-one teaching (one student to one teacher) is held as the gold standard in the field of education [36]. A very important part of this teaching is the conversational aspect between the student and teacher [66]. Using natural language to converse, the teacher guides and corrects the student, provides hints when he/she notices that the student is stuck, provides motivational support, asks the student questions, requires explanations from the student to see if they truly understand the material covered, and provides positive and negative feedback including insights into the material (e.g. via links, analogies, etc.). Often, good teachers can pick up on subtle cues before the student enters the impasse and can step in and provide guidance. Conversely, the student is able to talk with the teacher, helping the assimilation of knowledge.

Unfortunately in many computer-based systems, this very vital part of the learning process is missing. This is for good reason, as conversing on the same level as a human being is difficult to implement. However, there have been many attempts at making this a reality. When conversing is made possible, it is because, most commonly, the domain taught is restrictive and has

firm boundaries, i.e. when learning fractions, it is more likely that the conversations will revolve around the domain of fractions and not of some other topic such as politics. These parts of conversations that go back and forth between the student and tutor have been named "tutorial dialogues". Just as with human-to-human teaching, tutorial dialogues serve many purposes. Some ITSs have natural language capabilities whereas others communicate via sophisticated interfaces and hierarchies.

Why2-Atlas [81], AutoTutor [67] and Why2-AutoTutor [78] use the dialogues as the main activity in increasing the student's domain knowledge. The medical CIRCSIM-Tutor [114] and Geometry Explanation Tutor [8] (an extension of the PACT Geometry Tutor [6]) have problem solving as their main activity, but use tutorial dialogues to help remediate errors in student solutions while they are involved in the problem solving activity. The Shipboard Damage Control Tutor [64] waits until after the activity (a simulation) before engaging in a reflective debrief with the student. In the Mathematics DIALOG project[3] [31], natural language dialogues and subdialogues are given at particular points during the learning activity.

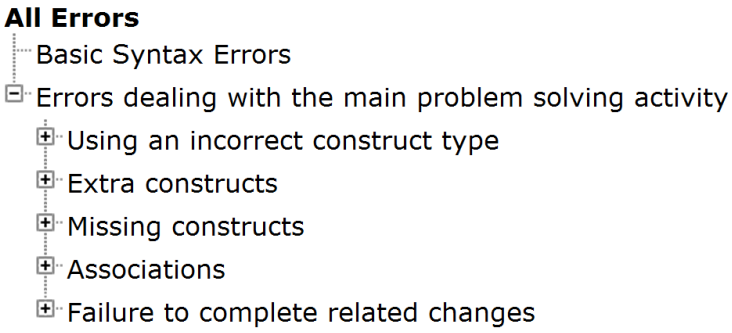

Figure 3.5: Overall view of the Revised Error Hierarchy [181]

As an example, we look at Weerasinghe and colleagues' work with tutorial dialogues [182, 181, 180]. They have used tutorial dialogues in a few ITSs, namely EER-Tutor, ERM-Tutor, and NORMIT, for error remediation. All these tutors provide a learning environment for students to solve problems. This selection of tutors covers both well- and ill-defined tasks and a

[3] http://www.ags.uni-sb.de/ dialog/

wide range in complexity of the problem solving tasks. Their model of using tutorial dialogues consists of three parts: 1) an error hierarchy, 2) the tutorial dialogues themselves - created for each error type, and 3) the rules for adapting the dialogues to each student. Although this model has been tried with constraint-based tutors, the researchers claim that it can be used with tutors that use other modelling techniques.

The *error hierarchy* is a tree structure of all the errors a student might make in the domain. The first three levels of the revised error hierarchy are shown in Fig. 3.5. The original hierarchy divided "All Errors" into "Syntactic" and "Semantic" errors. Syntactic errors are often simple errors to do with the language of the domain and so do not usually require a dialogue; the feedback from the tutor usually suffices. However, when the researchers generalised their hierarchy to other domains, they found that in some domains, more complex syntactic errors require a dialogue. Hence, they produced a revised version with "All Errors" divided into "Basic Syntactic Errors" and "Errors dealing with the main problem-solving activity". Looking at the error hierarchy from the leaves up to the root node, the leaves contain the atomic error (or errors) in its most basic form. For example, in constraint-based tutors, this would be the violated constraints. As one traverses up from the leaves to the root node, the error hierarchy generalises to more abstract concepts that are domain-independent; the top three levels are completely domain-independent. This is what allows the error hierarchy to be customised and used with any domain.

Tutorial dialogues are manually created for each error (i.e. each leaf node in the error hierarchy). An error signals that the student is having trouble with a domain concept. Therefore, that particular domain concept is targetted in the relevant tutorial dialogues. Each dialogue consists of four stages. The first stage informs the student of the associated domain concept and asks for justification of the student's actions. The second stage assists the student in understanding why the action is incorrect. The third stage helps the student understand how to correct the mistake. The fourth stage is a point of reflection, where the domain concept is re-enforced. Each of these steps starts a dialogue with the student, where the student is interacting with the system via fill-in-the-blank, true/false, or multi-choice type interactions.

The *rules for adapting the dialogues* are domain independent and enable individualisation of the dialogue by deciding on the timing, selection, and entry point into the dialogue. The current student model informs these decisions. As an example, rule 1 deals with the timing of initiating a dialogue. If the student has been inactive for a certain period of time, this model in the ITS initiates a dialogue with the most suitable error in their current (sometimes, unsubmitted) solution. This infers that the student is struggling with the results of that particular error and is finding it difficult to move on. As another example, rule 3 deals with the selection of the particular tutorial dialogue. Often when students submit incorrect solutions, there are multiple errors. Instead of dealing with all errors at once, rule 3 selects the most appropriate error; this is informed by a number of things including their student model.

3.2.7 Learning by Teaching

The form of learning by teaching amongst humans is often referred to as "peer tutoring". Peer tutoring has long been seen as a valuable exercise which benefits both the tutor and tutee, while lowering costs and providing a large pool of tutors [9]. Peer tutoring is different from peer collaboration or other forms of peer learning. In peer tutoring, one person has been designated to teach, while in peer collaboration or other forms of peer learning there are no asymetric roles [147]; the roles of tutor and tutee swap continuously. With peer tutoring, researchers have shown learning gains not just for non tutors but for tutors as well [49]. This is called the tutor learning effect [147]. However, these gains depend on a few factors [147]:

- If the tutor's teaching method is by knowledge telling (knowledge transfer from tutor to tutee), the tutor learning effect is minimised. Instead, if the tutor's teaching method is via knowledge building (using explaining and questioning), then the learning gains are high for both tutor and tutee. When the tutor has to generate explanations and ask questions about concepts, it forces the tutor to think and reflect on the material that is being taught.

- If the tutor asks shallow questions or gives textbook explanations, learning is minimised.

- Learning gains could be reduced if the tutor and tutee are around the same age; the tutor might not always act as a tutor and tutoring then often becomes reciprocal. Cross age tutor/tutee relationships keep the assymetric effect.

- Conversely, if there is a large age gap between the tutor and tutee (which might mean a large domain knowledge gap), the learning gains for the tutor might be reduced if they are teaching far below their level. However, tutor learning effects have been found at most age groups.

- Some students are better than others at self-explaining. Being able to self-explain well has a large effect on learning. Tutees who self-explain well might reduce the tutor's learning gains.

- Tutors from minority or disadvantaged groups seem to have a higher learning effect.

Learning by teaching is a complex strategy (one that incorporates other strategies). It depends on several strategies working together to maximise the tutor's learning. For example, a tutor with good self-regulating skills will do far better than a tutor without. Self-regulating skills can be learnt; however, to have a computer-based system that promotes self-regulated learning by teaching is a difficult ask. The tutee's knowledge must be transparent to the tutor and the tutor must be able to build on this knowledge via knowledge building.

An example of computer-based "learning by teaching" is a system that Biswas and colleagues from the Teachable Agents Group have implemented. The system is called *Betty's Brain* and the effectiveness of the system has been researched in classrooms. Betty's Brain [33] is a teachable agent used to support middle school children as they learn about river ecosystems. Teachable agents are computer-based agents (that represent simulated students)

placed within a learning environment to support learning by teaching. Instead of instructing a human peer, students teach Betty, the computer-based agent, about various concepts and causal relationships by drawing a concept map which represents Betty's knowledge at any one time. For example, students might learn about two concepts, "fish" and "carbon dioxide" and relationships like "fish exhale carbon dioxide". They can also enter whether the first concept causes a change (increase or decrease) in the second; for example, fish exhaling carbon dioxide increases carbon dioxide in the river [148]. The Betty's Brain interface can be seen in Fig. 3.6.

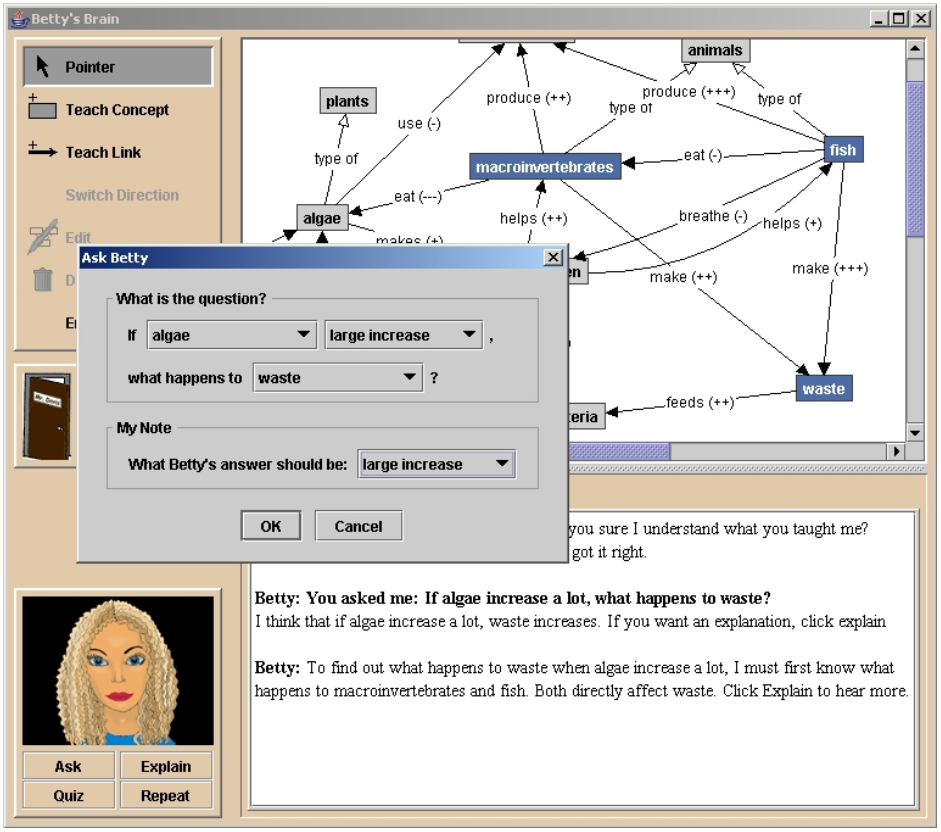

Figure 3.6: Interface of Betty's Brain [80]

There are three methods or activities that students partake in, when using Betty's Brain: teach, query, and quiz [33]. During *teach*, the students create and modify a concept map; Betty comes with no prior knowledge. During *query*, the students can ask Betty questions regarding what they have

taught her. Betty's uses the concept map to follow chains of concepts and relationships to answer a query. She can only do this from the concept map that the student has drawn i.e. from what the student teacher has taught her. The *quiz* action allows Betty to take a quiz of pre-defined questions. The questions are then marked by Mr. Davis, a mentor agent. The results of the quiz are a reflection of the student teacher's knowledge. The student can then re-learn the information, modify the concept map, and try the quiz again.

In the newer, self-regulated learning version (Betty's Brain SRL) [34, 32], Betty's persona incorporates metacognitive knowledge (in the form of motivational and self-regulation cues) that she conveys to students when it is triggered. Mr. Davis (the mentor), whose responses are also triggered by the student's activity focuses on self-regulation strategies. For example, if a student asks Betty many questions without requiring her to explain her answers, Betty will ask the student to make her explain the answers to make sure she understands the concepts more fully. Similarly, Mr. Davis will prompt the student to click on the "explain" button and ask Betty to explain her answers. Betty's Brain SRL has a number of such regulation goals. It monitors the student's knowledge by asking the student to ask Betty for explanations, it tracks the student's self assessment, it tracks the student's progress (by tracking Betty's scores in the quizzes), and it helps to set learning goals (triggered when Betty cannot answer a question)[34].

3.2.8 Problem Posing

Problem posing is another complex strategy. It has been primarily used and researched in the Mathematics domain [157, 158, 156, 59]. Problem posing refers to both the generation of new problems and the re-formulation of given problems [156]. Problem posing can be done either as the main goal of the exercise or as part of problem solving (often subconsciously).

With active problem posing (i.e. when the goal is to pose a problem), students are given some background knowledge or some constraints and asked to come up with a problem within those constraints. The type, complexity, and variations of the problems say something about the student's expertise

level, particularly about their level of critical thinking and analysis. Automatic problem posing (when the main goal of the exercise is not to pose a problem) can occur while a student is trying to solve a particular problem. They might reformulate the problem as a different type of problem or one that has different attributes so that they can then see the original problem in a different light and solve it from a different perspective. In fact, the difference between novices and experts while solving problems is the amount of time spent formulating and reformulating the problem; experts spend far greater time in such qualitative analysis [155].

Problem posing was first coined as a phrase and used as a pedagogy by Paulo Freire, a Brazilian educator. In his book, "Pedagogy of the Oppressed" [63] he talks about both the traditional educational pedagogy of that time and the one he proposes. He uses the term "banking education" to portray the education of the day, where students are like empty bank accounts ready to be deposited with knowledge by an external entity. The students themselves do not have control over this knowledge and are not encouraged to think of its validity or accuracy. This, he says, leads to an oppressive society, where the oppressed simply take on the knowledge of the oppressors. Instead, he proposes a different type of education called Problem Posing, where given a certain amount of knowledge, the student comes up with related problems to expand their knowledge. Unlike problem solving, the emphasis here is not on the solution (one does not even need to have a solution in this case), but instead in the way the problem is formed. With this pedagogy, students and teachers are seen on an equal footing (eliminating social inequality), being co-creators of knowledge. He argues that this type of learning enhances the student's critical thinking and analysis skills.

Even though problem posing is seen as a valuable strategy, it is not a popular one. This is understandable, as assessing students using the problem posing strategy is a big task [73]. The scope of problems posed is very large. The teacher needs to check each problem posed and assess it based on several criteria, ones that are not easy to assess, before providing individual feedback on each problem. Students could make problems that are simply incorrect, repeatedly make similar problems, or make problems that are too simple to be useful [70]. This is where a computer-based system that could assess the

problems would be very beneficial. However, the difficulties faced by normal problem posing assessments are compounded when implemented within an ITS. The ITS needs to provide students with a way to formulate the problem structure. Furthermore, it also needs to be able to deem the quality and accuracy of the problem posed.

Problem posing is usually categorised into three types: 1) Problem-based, 2) Story-based, and 3) Solution-based [132]. In *problem-based problem posing*, the student is presented with a problem and asked to create problems that are similar to the original problem and have the same solution. For example, if the original problem shown was:

$$1 + 1 = 2$$

then, they could create multiple problems such as the trivial ones shown below:

$(1 + 1) + 1 = 3 - 1$ or
$(1 + 1) + (1 - 1) = 2$

In *story-based problem posing*, the student is given a realistic story and the student has to pose problems relating to that. For example, a part of a story might say:

> "William travelled 20km to meet his father. William's brother, Danny travelled 40km to meet his father."

Some trivial problems that might be posed from a story containing such a part might be:

> "How much further did Danny have to travel than William?"
> "Why did Danny travel twice as far as William?"

Not all problems might be able to be solved with the information given. However, the exercise is in formulating the problem with the given information, which in turn extends the amount of knowledge.

With *Solution-based problem posing*, the student is given a method to solve the problem. The student must formulate a question that has a similar solution method to the original. For example, a solution method might be given, such as:

$$5 + x = 8$$

The student then needs to construct problems that have the same solution method. The problem could be a word problem, such as:

> "Andrew has 5 apples, but he needed 8. Sam gave Andrew enough apples. How many apples did Sam give Andrew?"

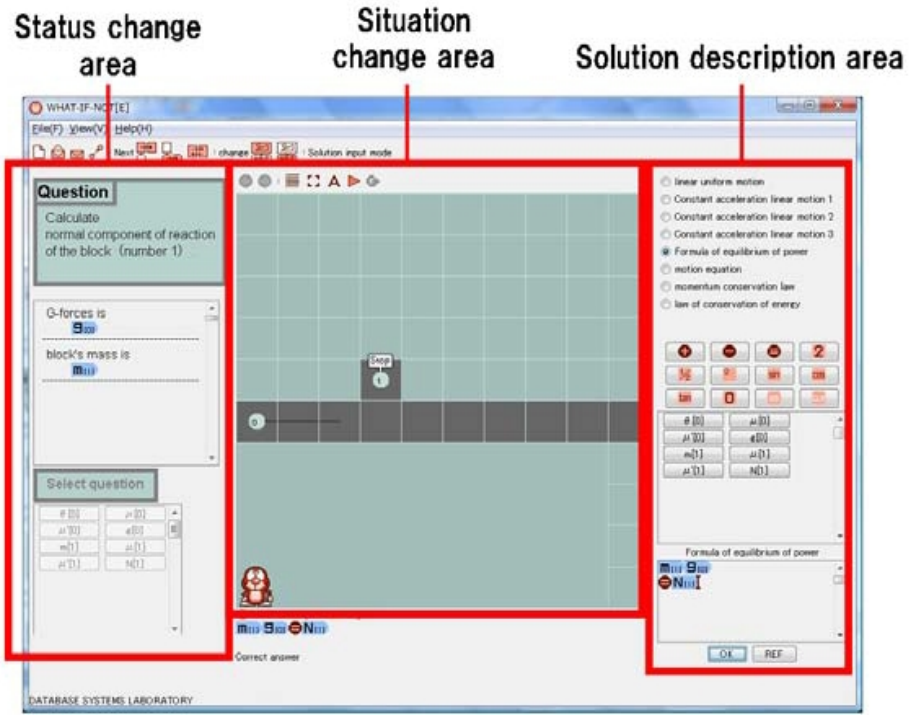

Figure 3.7: The Problem Change Interface [188]

Most of the work of problem posing in computer-based systems has been carried out by Hirashima and colleagues. They have created a number of Intelligent Learning Environments (ILEs), in the domains of early arithmetic and early mechanics (Physics).

The interface of the mechanics problem posing tutor is shown in Fig. 3.7. This tutor is a problem-based problem posing tutor [188]. The original question is given in the top left of the screen. The student can change the original problem by changing the physical objects in the situation change area. For example, they might drag the block to a different position. These changes affect a set of attributes. In the status change area, students can then specify the status of the attributes to either given, unknown, or required. Once they have done that, they then complete the solution using the solution description area. After completing the solution, they are presented with a comparison screen, where they can compare the original problem and solution to theirs.

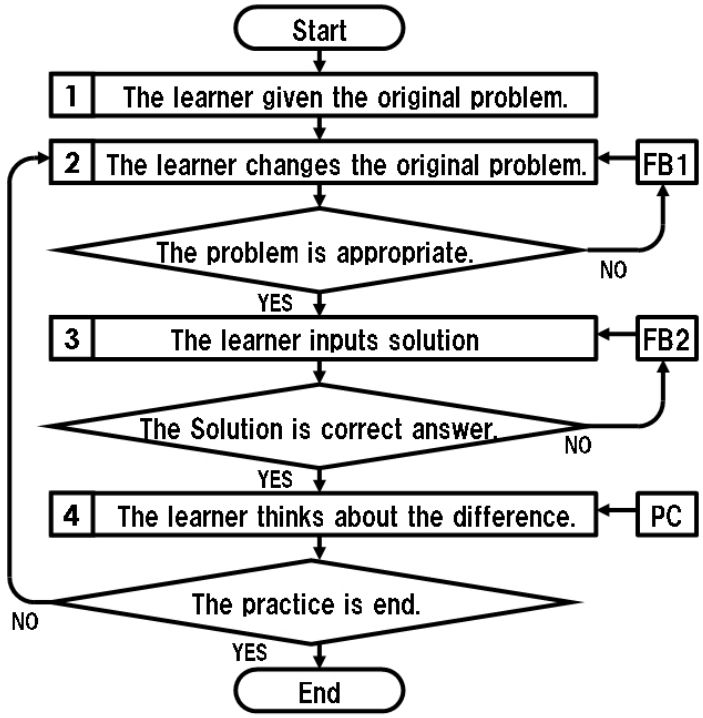

Figure 3.8: The flow of practice of Problem-based problem posing [188]

Fig. 3.8 shows the flow of events while working on this tutor. First, the learner is given a problem, which they then change. If the problem is deemed appropriate, the learner inputs the solution, otherwise relevant feedback is provided and the learner is given another chance. If the solution is correct,

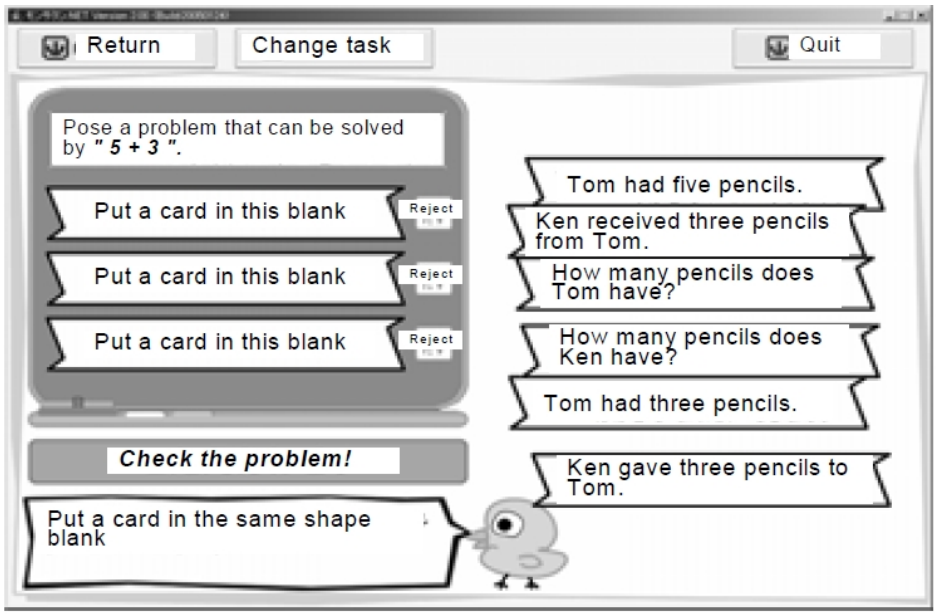

Figure 3.9: The Monsakun interface [73]

the learner is presented with the problem comparisons, so that he/she can reflect on the differences.

An example of a solution-based problem posing interface is shown in Fig. 3.9 [73, 71]. In this tutor, a solution method is provided at the top left of the screen. A number of sentence structures relating to concepts are available on the right hand side of the screen. The empty solution space is on the left of the screen. A student has to create a word problem by dragging and dropping the appropriate sentence structures into the slots in the solution workspace. As the cards are dropped in their slots, they are either rejected or accepted depending on their ability to form correct problems. Examples of correct and incorrect problems relating to the interface in Fig. 3.9 are given below.

The correct problem: The solution method is given at the top left as being "5+3", i.e. the student's problem should reflect this structure. A correct way of doing this would be for the student to place the following three cards from the right hand side of the interface into the slots on the left hand side in this order:

- Tom had five pencils.

- Ken gave three pencils to Tom.
- How many pencils does Tom have?

The incorrect problem: Again, the problem statement is given (5 + 3). One incorrect problem (from a number of incorrect problems) the student could make would be to place the cards from the right hand side into the slots on the left hand side in this order:

- Tom had three pencils.
- Ken received three pencils from Tom.
- How many pencils does Tom have?

The details of the diagnosis function can be found in [70]. Experimental analysis of these tutors have shown that they are both useable and effective [72].

3.3 Multiple Pedagogical Strategies in ITSs

Unfortunately, not many researchers have attempted work in the area of having multiple strategies in ITSs for each learning point. Mizoguchi and colleagues have designed a system to help student teachers prepare their lessons [84]. This was done by first creating an ontology of teaching theories called OMNIBUS. Using OMNIBUS, they built the SMARTIES project, which enabled teachers to interact with the ontology and in return, get tutoring actions based on these theories [84, 126]. Using the ontology, users can choose one of a few pedagogical theories and receive the tutoring actions based on that.

ITS authoring systems have been used to make the creation of ITSs much easier, some to the point where no programming skills are required to create a basic ITS. These ITSs however, have some assumptions. They might have assumptions on how the domain is modelled. For example, CTAT[4] [7] assumes model tracing whereas the ASPIRE Authoring System [124] assumes

[4] http://ctat.pact.cs.cmu.edu

constraint-based modelling. Pedagogical strategies at points of learning are either assumed (and automatically added) or sometimes the teacher has some control over certain aspects of the pedagogy. For example, in ASPIRE, teachers can choose how many feedback levels (and the content of the feedback levels) to provide at various points (steps) in the tutoring process. However, none of these still give the flexibility of fine tuning each strategy for each of the learning points.

The main amount of work in this area has been done by Ainsworth and her colleagues within their system REDEEM.

Reusable Educational Design Environment and Engineering Methodology (REDEEM) [3] is an ITS Authoring Tool (ITSAT), developed primarily so that teachers can easily add a set number of pedagogical strategies around domain content (which might already exist in the form of a CBT), thus providing some form of adaptation to groups of students. REDEEM is a great example of trying to make the addition of pedagogical strategies to an ITS easier for teachers who are not computer experts. With REDEEM, the authors of the ITSs are the teachers themselves. Another newer example of an ITSAT where teachers or domain experts are also authors is the ASPIRE Authoring System [124].

The REDEEM suite consists of three main pieces of software: courseware catalogues, authoring tools, and an ITS shell. Using these three pieces, the domain can be structured and adapted to groups of students with the addition of interactive pedagogy.

In REDEEM, domain content is seen as pages. Each page could contain text or multimedia. Pages can be grouped together into sections. Sections do not have to have sequential pages. Creating sections could be based on some other factor, such as difficulty level, choice of topic, or any factor that the author chooses. This means that pages in a particular section are similar to each other in some way. Authors then describe relations between sections.

Relations define any type of relationships that might exist between sections. A common relation is the *pre-requisite* relation, where a section can only be offered to students after all the pre-requisite sections have been completed. These relations between pages and sections are what the ITS shell use to provide default adaptation.

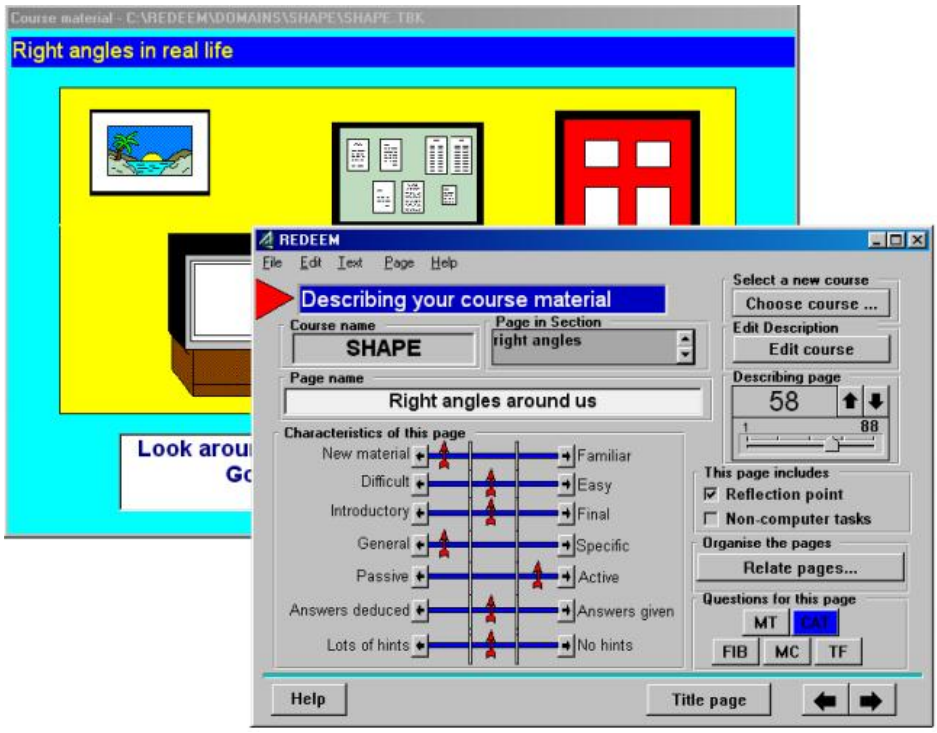

Figure 3.10: Authoring domain content in REDEEM [1]

As an example, the interface for the content description process can be seen in Fig. 3.10. Here, a teacher has created a course name called "Shape" and is currently describing the page "Right angles around us". In the figure, the page is sitting behind the description interface. The teacher then has seven sliders, which they could place in one of three positions. For example, they have described that this page contains *New material* with *Average* difficulty. They have also said that this is an *Active* page (the student is required to interact), rather than a *Passive* one. Using the same interface, teachers can add interactivity to the page. There are two main methods of interactivity: one non-computer based, and the other, computer-based. For the non-computer based tasks, each page could contain one of two non-computerised tasks: a reflection point or other non-computer tasks such as asking the student to take personal notes. This teacher has specified that the students will be required to go through a reflection point, which simply might be a point where they stop and reflect on what they have learnt so far. For the computerised tasks, the teacher can also choose to attach a question to

this page. There are five different types of questions that could be authored: Multi-choice Multiple true answers (MT), Matching (MAT), Fill in the blank (FIB), Multiple choice (MC), and True/False (TF). The teacher could also input up to five levels of help for each question. Using this method, teachers describe each page in their courseware content.

Students use the system which is served by the REDEEM ITS shell. Students are divided into a number of author-specified categories. Categories can contain as many or as few students as the author decides. Similar to the idea of sections, students in the same category have something in common. For example, the author could have divided the students by their performance level i.e. students in a particular category perform similarly to their counterparts in the same category. The shell could later place students in a different category (if the teacher allows it) depending on their performance on questions. This is determined by their student models, which REDEEM keeps up to date for each student. Being in a different category could now result in the student being faced with a new pedagogical strategy.

Next the author creates one or more teaching strategies. They do this by altering the position of the sliders and ticking the checkboxes provided through the mechanism to define their strategy (see Fig 3.11). With these combinations, up to 10,000 alternative strategies that are subtly different to each other can be produced, although no author had created more than seven [2]. At one extreme, the teacher could create a strategy that allows the student full freedom with the ability for discovery learning. At the other extreme, they could create a guided strategy, where the student has a set path and must complete the questions in the correct order and move on only if they get them correct.

In the final stage of authoring in REDEEM, authors choose a strategy for each category of students. The authors base their decisions on their knowledge of the type of students in each category and the type of teaching strategy they want them to encounter. This type of strategy application is defined as macro level adaptation, where the system responds to changes in the environment, as opposed to micro adaptation where the system responds to anything the user does within the system. ITSs generally rely on micro adaptation, with the ability to recognise and respond to each user input. The

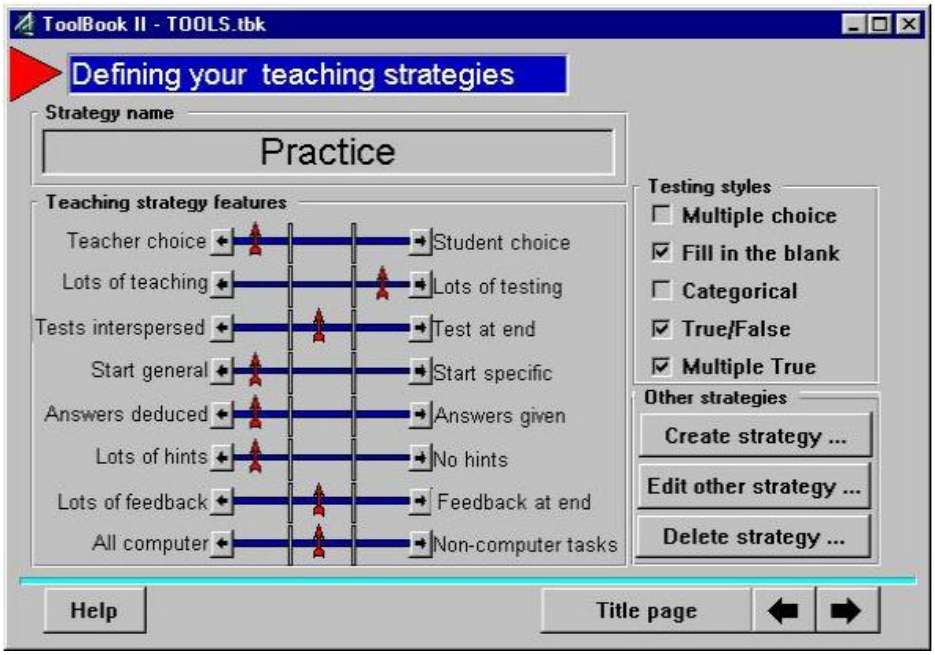

Figure 3.11: Authoring teaching strategies in REDEEM [1]

MAPS Framework consists of an ITS that responds at the micro adaptation level.

REDEEM has been evaluated in classroom situations [1] and in a Naval Academy setting [4] with promising results. With respect to authoring, authors (who had no prior experience developing a CBT) have taken between one and five hours to create an hour of REDEEM instruction from existing CBT [1]. The developers of REDEEM believe as we do, that these types of environments should exploit the symbiotic relationship between psychology and authoring environments [100]. Although the MAPS Framework is not an authoring system, we believe that there is a strong relationship between the pedagogical strategies entered into MAPS, education, and psychology and that one dimension helps foster the other.

Chapter IV

Exploring the Building of a Strategy and the Prototype of the Framework

This is the first major stage of our research. We wanted to see if we could, by exploring large datasets, produce a pedagogical strategy, which may or may not be an ideal strategy (from an educational viewpoint), but one that is ready for experimental testing. Looking at large datasets and analysing them is called Educational Data Mining (EDM). EDM is a relatively new and growing field where analysis of large datasets of student-computer interaction logs are used to answer educational research questions [22]. It is "an emerging discipline, concerned with developing methods for exploring the unique types of data that come from educational settings, and using those methods to better understand students, and the settings which they learn in" [77, 146]. Data mining usually occurs post-hoc (after the intended experiment or learning session).

EDM is used quite frequently nowadays by both ITS and statistical researchers. ITS researchers are generally interested in finding out how students actually behaved after their experimental study/learning session and to delve deeper to find any significant correlations between variables. These could then be used to manipulate the independent variable over a number of studies to pinpoint the best range of values which would produce the best gain in learning. Statistical researchers are usually interested in building models of the certain behaviours and using those models to predict the probability of future behaviour of a student, given certain conditions. In some cases, the results of datamining are fed back and used in either directly affecting the ITS or in helping modify it. Much work in this area has been done on the Assistments Tutor, a tutor which assists students while assess-

ing them [173, 62]. The CanadarmTutor, an intelligent agent which teaches students to solve problems using a robotic arm in a complex virtual environment harnesses this ability. Its framework consists of an extended sequential mining algorithm that is used to extract patterns and relationships from the behaviour of human experts executing the same task [134, 133].

We, however, do not want to do either. We were interested in an earlier step i.e. could educators (with the help of data miners) build their own potential pedagogical strategy based on sufficient evidence from data mining? That is our first step. If so, could they then enter that strategy into any ITS (built according to our specifications) and adjust the variables accordingly to test that strategy in an experiment? This would then answer our question about the possibility of having such a framework for ITSs.

4.1 Exploring Help-Seeking behaviour in ITSs using SQL-Tutor

The results of this section have been published in [103].

There are usually multiple ways of getting help in ITSs. Not all methods are implemented in any one ITS and each method gives a different type of help to the student. Help in an ITS can be classified into tutor-specific or domain-specific. Tutor-specific help shows the student how to use the tutor itself; it is specific to each tutor. This includes tutorials on how to use the solution workspace, how to navigate to the next problem, or how to submit a solution. In contrast, domain-specific help concentrates on concepts within the domain, irrespective of the tutoring system used. This is analogous in the human tutoring world, where tutor-specific help is unique to each human tutor and includes instructions on how to get more help (e.g. lab schedules, tutor's email address or office hours), how to submit solutions to the labs, or instructions on how to get access to and use any apparatus. Domain-specific help is help the tutor gives about the domain itself. While two separate human tutors might differ in their tutor-specific help, the general content of their domain-specific help should be similar as they relate to the particular domain. In this section, we only concentrate on domain-specific help.

Domain-specific help can be categorised in many ways. One way is to class it into problem-specific, solution-specific, or concept-specific help. Problem-

dependent help is related specifically to the problem at hand. Problem-specific help could be hints or feedback messages provided for a particular problem or step within that problem. For example, in SQL-Tutor, to help the students with a problem, the schema of that problem's relational database is presented in an easy-to-understand layout. Clicking on each item in the schema could bring up more details about that item in context to the problem. Thermo-tutor [125] provides automatic calculation or remember-vs-recall type help for many of the steps that are not directly related to the learning goal. Andes [176] provides a whole array of help and calculators that can help the student understand and interact with the problem better. Some problem-specific help could be given before the problem has been attempted (e.g. forward hints) or after a solution has been submitted (feedback).

Solution-specific help is help given to the student regarding their own solution. This type of help is generally given after the student has submitted a solution or part of it. For example, in SQL-Tutor, part of the feedback might say "You are missing the SELECT clause in your SELECT statement". This feedback is directly referencing and is concerned only about the solution that the student submitted.

Concept-specific help is help on the domain concepts itself. This helps the student focus on areas within the domain that the student needs to learn to be able to solve the problem. As an example, when a student makes an error in the SELECT clause in SQL-Tutor, on certain help levels, it gives concept-specific help, explaining the use and meaning of the SELECT clause. Concept-specific help could be given at any stage during the session; it could be given as tutorials on a particular concept (e.g. Andes), or as generalised feedback on submitting a solution or step in a solution (e.g. NORMIT). For example, the use of the Error Hierarchy in EER-Tutor, allows the feedback to move from a solution-specific help to a concept-specific help with the use of tutorial dialogues [182].

Domain-specific help could also be classified according to its adaptivity. Adaptive help is based on the student's individual model and provides specific help for that student based on their competency (either using historical or current evidence from their student model). An example of a non-adaptive help is the full solution.

For this research, we categorised domain help into low-level help (LLH) and high-level help (HLH). LLH and HLH refer to the degree of help the student gets each time. LLH implies that they get less domain specific help (e.g. "you have made errors in your SELECT statement" as opposed to HLH which might provide them with a partial or full solution). The closer the tutor helps them achieve a correct solution with its help, the higher the level of help provided. Sometimes, the process of giving higher degrees of help until the full solution for that step (or problem) is given is referred to as "bottoming out". In this section, we explore the use of LLH and HLH and its effect on learning.

4.1.1 Domain help in SQL-Tutor

In SQL-Tutor, help is given on a student's solution submission as feedback. There are six numbered categories of help (see Fig. 4.1). These in order of degree of help are: 1. Simple feedback, 2. Error flag, 3. Hint, 4. Partial solution, 5. List all errors, and 6. Full solution. The degree of help (also known as the feedback level) is controlled on the interface by a simple combo box. When a student begins a new problem, the help level defaults to 1 (the lowest degree of help). On subsequent incorrect solutions, the tutor automatically increments by one the help level to a maximum of 3 (i.e. hint). The student however, can override this help selection at any time by simply manipulating the value of the combo box. Help levels 4-6 have to be requested by the student explicitly (i.e. they do not automatically increment past level 3). For this research, we have thus classified help levels 1-3 (inclusive) as LLH and 4-6 (inclusive) as HLH. In Fig. 4.1, the student has requested HLH (in this case, the full solution) for the problem, which is displayed in the feedback pane.

4.1.2 The data and the methodology used

There are many versions of SQL-Tutor available. One such version has been online and is available to students worldwide who have purchased certain database books[1]. We downloaded student data (models and logs) from this

[1] From Addison Wesley Publishers: http://www.aw-bc.com/databaseplace/

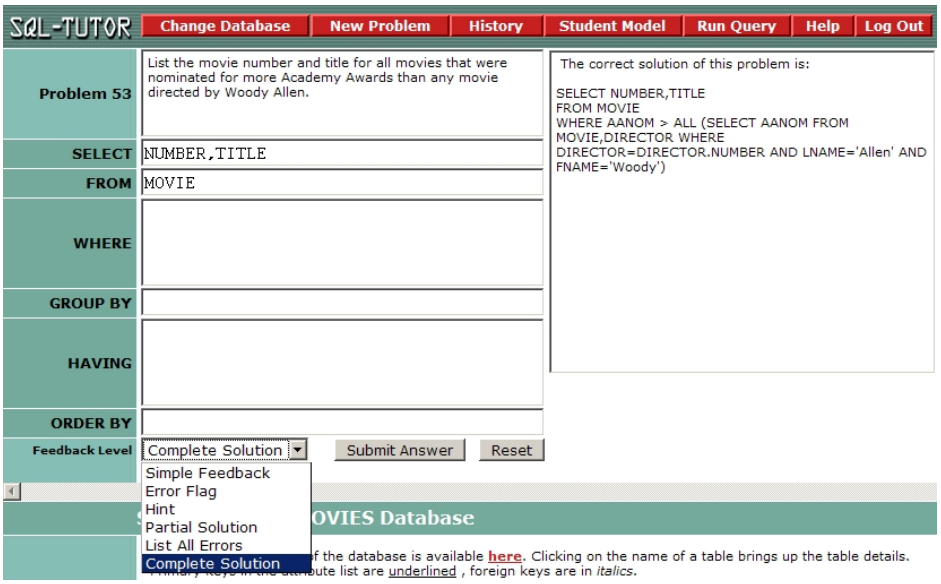

Figure 4.1: The SQL-Tutor task environment showing the various levels of help.

version of SQL-Tutor and excluded any student who made fewer than five attempts. The number five was chosen as a cut-off point because 1) it would give us data on students who had made substantive attempts using SQL-Tutor (remember that one attempt is not simply a step, but could be an entire solution in SQL-Tutor), and 2) using the tutor normally, it would take at least 5 attempts to realise that the HLH has to be chosen manually. The remaining data consisted of 1,803 users who had made a total of 100,781 attempts and spent just over 1,959 active hours on the system. Active hours consists of time spent actively solving the problem; not just session times. Session times are calculated from the moment the student logs in to the moment they log out (or are automatically logged out). Session times include a lot more time spent (particularly nearing the end of the time) when students are not engaging with the system. Students in this dataset on average solved 70% of the problems they attempted; giving a grand combined total of 19,604 solved problems. We called this the main dataset (dataset A) as we performed the same analyses on two significantly smaller datasets (datasets B and C) based on data collected at the University of Canterbury during laboratory sessions in a second-year database course. As the results for all three datasets were very similar, we concentrate on dataset A here.

While looking at each attempt more closely, we found that not all subsequent attempts were different to each other. Some incorrect attempts were valid, while other attempts that subsequently followed, contained exactly the same solution. It is as though the student submitted the solution again, expecting some different output. We figured that these repeated same attempts were in fact the student asking the system for more help. In other cases, we sometimes found the student submitting empty solutions. These empty solutions were also sometimes immediately followed by empty solutions submissions again. We figured that a student who submits an empty solution is not expecting the system to give them feedback for their solution; instead they are perhaps hoping the system would provide them with some help. Submitting the empty solution again was simply requesting more help. We coined the phrase *Requests for Help (RFH)* and classed these behaviours as such. We hypothesised that sometimes quick successions of repeated submissions might be accidental or due to other non-student related behaviour such as network errors. Taking this into account, we only counted successive attempts that were at least one second apart. We thus defined a valid solution to be a solution (whether correct or incorrect) that is different in some way to the previous solution to the same problem. We also defined RFH as either an empty solution or a solution where the student has made no changes between the current solution and the previous solution submission. With RFH, the student is hoping that more hints or feedback will be provided with each submission. SQL-Tutor automatically increments the help level by one up to a maximum of 3 for each incorrect submission. Equation (4.1) shows the relationship between submissions, valid attempts, and RFH.

$$Submissions = Valid\,attempts + Requests\,for\,help\,(RFH) \tag{4.1}$$

We further categorised the valid attempts into LLH or HLH attempts, giving us Equation (4.2) and then deduced the number of HLH attempts for each student.

$$Submissions = LLH\ attempts + HLH\ attempts + RFH \tag{4.2}$$

To normalise the HLH attempts over all the students, we created an HLH ratio such that $0 \leq HLH \leq 1$ (see Equation (4.3).

$$HLHratio = \frac{Number\ of\ high\ level\ attempts}{Total\ number\ of\ attempts} \tag{4.3}$$

This ratio categorises students based on their use of HLH and makes it possible to compare students' values with each other. To explain the the ratio further, a student with a ratio of 1 uses HLH (e.g. full solution) for every attempt, while a student with a low ratio, say 0.01 uses HLH on average 1% of the time.

To calculate how frequently HLH is used, we ordered students according to their HLH ratio (i.e. according to their use of HLH) into ten groups (A1-A10); from low frequency HLH users to high frequency HLH users. The frequency graph is shown in Fig. 4.2. The ten bars on the graph represent groups A1-A10, where A1 has an HLH ratio of 0.05. Each group was further divided into two, to provide us with a more accurate description of the state of affairs within each group. The need for this further breakdown will become clear shortly, when we discuss group $A1$; group $A1$ seemed to be made up of two distinct subgroups.

Student models and logs were analysed from each of the groups; learning curves were also graphed for each group.

4.1.3 Results and Discussion

Very similar results were found for the other datasets (B, and C). To avoid repetition, we only discuss results from our main dataset (A) here.

Frequency distribution of users according to their HLH

The frequency distribution graph shown in Fig. 4.2 shows the number of students grouped according to their usage of HLH.

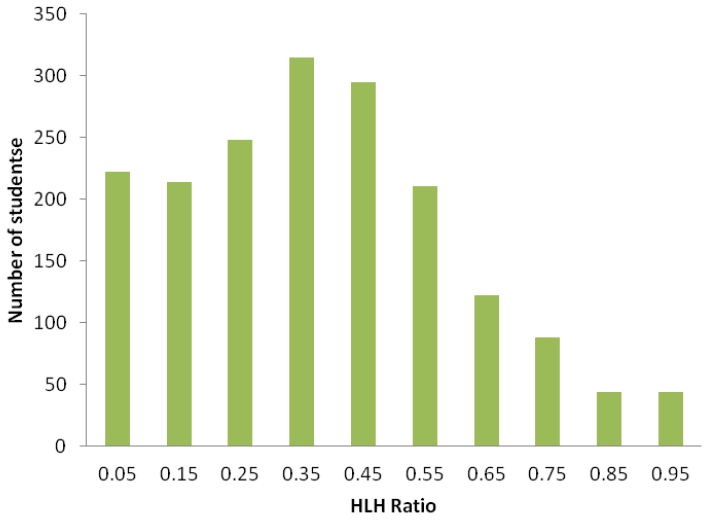

Figure 4.2: Frequency distribution of users in the ten HLH user groups in dataset A

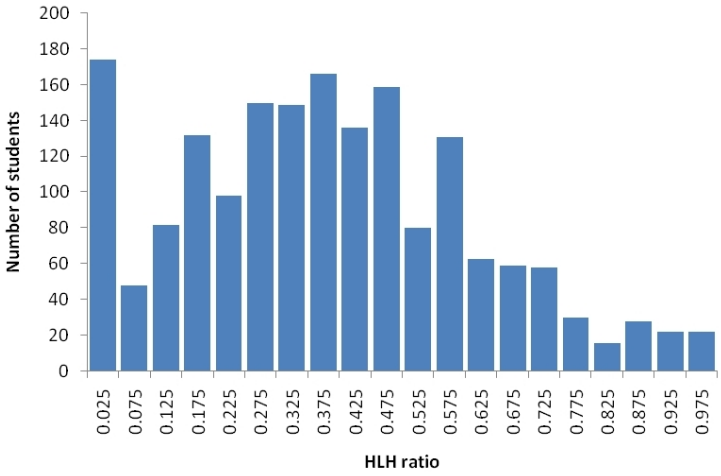

Figure 4.3: Frequency distribution of users in the twenty HLH user groups in dataset A

This frequency graph has an interesting shape. From $HLH \geq 0.1$, it is a positively skewed normal graph. However there a peak external to that part of the graph at the beginning in $A1$ ($HLH \leq 0.01$). Students in this part of the graph are very low users of HLH. So, why is there such a large number of users in this group compared to the rest of the graph? Each of the ten groups (A1-A10) were further divided into two groups (e.g. $A1$ is divided into $A1_1$ and $A1_2$). This was done as we realised that there was something interesting going on in the group $A1$. This frequency graph (with the twenty groups) is shown in Fig. 4.3.

Manual inspections of the logs and models indicated that group $A1$ was in fact made up of at least two distinct subgroups, $A1_1$ and $A1_2$. One subgroup (group $A1_1$) consisted of students who had higher domain expertise than anyone else in the population. This group did not need to use HLH to get their answers correct as they already had higher declarative domain knowledge. The other subgroup ($A1_2$) had low domain knowledge, but insisted on using low amounts of HLH. Further experimentation needs to be done to fully understand this behaviour. For example, these students could have low meta-cognitive (help-seeking) skills or, due to social norms, feel that looking at HLH constitutes a form of "cheating".

Learning curves for each HLH group

Learning curves were calculated and drawn for each of the ten groups ($A1$-$A10$). Fig. 4.4 shows the learning curves all plotted on the same graph for ease of comparison. To avoid cluttering the graph, the equations have been excluded from the graph, but are included in Table 4.1.

There are a few interesting things to note about these graphs.

1. The graphs follow an approximate ordering according to their HLH. The higher the HLH use, the shallower the learning. For example, the students in $A10$ who are extremely high users of HLH have a very shallow learning curve i.e. the probability that they will continue making errors is still quite high, even as their learning rate approximates zero.

2. The fit for the power curves is quite high across all groups. The lowest

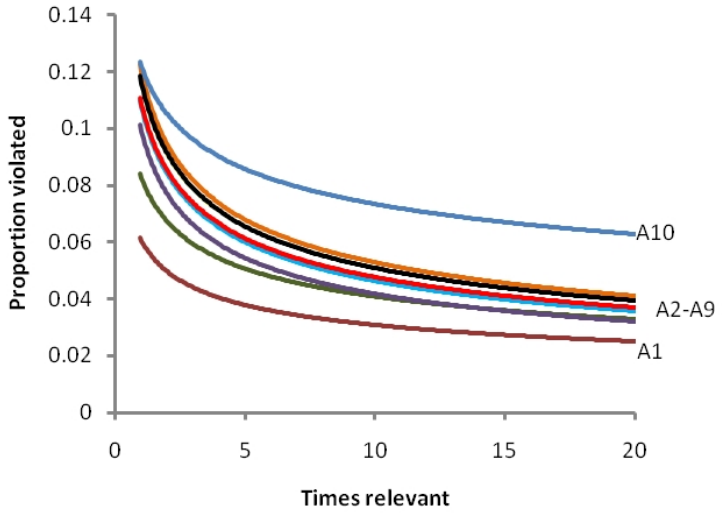

Figure 4.4: The learning curves for the ten HLH user groups ($A1$-$A10$

Table 4.1: Power curve equations and fits (R^2) for the ten HLH groups ($A1 - A10$) in SQL-Tutor.

Group	HLH ratio	Power curve equation	R^2 fit
A1	0.0 - 0.1	$y = 0.061x^{-0.30}$	0.844
A2	0.1 - 0.2	$y = 0.084x^{-0.31}$	0.956
A3	0.2 - 0.3	$y = 0.101x^{-0.38}$	0.955
A4	0.3 - 0.4	$y = 0.019x^{-0.37}$	0.956
A5	0.4 - 0.5	$y = 0.110x^{-0.36}$	0.965
A6	0.5 - 0.6	$y = 0.123x^{-0.39}$	0.961
A7	0.6 - 0.7	$y = 0.122x^{-0.36}$	0.953
A8	0.7 - 0.8	$y = 0.115x^{-0.35}$	0.953
A9	0.8 - 0.9	$y = 0.118x^{-0.36}$	0.912
A10	0.9 - 1.0	$y = 0.123x^{-0.22}$	0.956

fit for curve is in $A1$ ($R^2 = 0.844$). Although this is still high, it stands out from the other fits for curves. This could be because, as mentioned earlier, this group contains two distinct subgroups ($A1_1$ and $A1_2$). R^2 usually attempts to describe transferability of skills. The lower overall fit could thus mean that $A1_1$ and $A1_2$ have quite different transferability skills.

3. The learning curves for the middle portions ($A2 - A9$) are very similar (e.g. in learning rates, depth, etc). This leaves us with three distinct groups: $A1$, $A2 - A9$, and $A10$. In our dataset C, this distinction was slightly different, but still very similar $A1$, $A2 - A8$, and $A9 - A10$. We have plotted these categories on a separate graph 4.5 showing that the "middle" groups show similar types of learning while the extreme high and low HLH users display markedly different learning.

4. The χ value increases as HLH use increases. The χ value is an indication of how difficult the students find the domain material. There is also a marked difference between the χ value of group $A1$ and the other groups, possibly indicating the fact that this group contains at least some experts or students with prior domain knowledge.

5. The intermediate group ($A2$-$A9$) have similarly high learning rates. The lowest learning rate (0.22) belongs to the group using the most HLH ($A10$). This could be because students relying heavily on HLH do not necessarily engage with the material compared to students using lower amounts of HLH. This has detrimental effects on their learning. For learning to be effective, it requires active engagement with the material for construction of their internal representation of knowledge (active practice rather than passive [43]), deliberate practice [61], and the ability to learn from their own errors [137]. The other group that has lower learning rates is $A1$. Group $A1_1$ (the experts) already have a higher knowledge to begin with and therefore their learning curves are not as steep as they approach their asymptote of learning quickly. The second subgroup ($A1_2$) perseveres to solve problems without utilising HLH and therefore has a slower learning rate than the other groups.

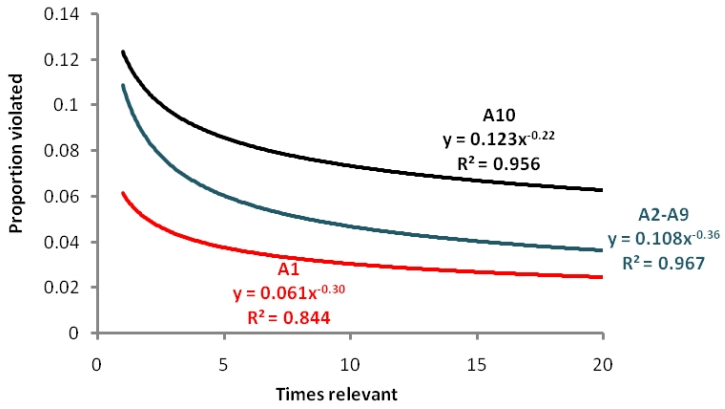

Figure 4.5: The learning curves for the HLH user groups ($A1$, $A2$-$A9$, $A10$

These two reasons could show why group $A1$ has a lower than expected learning rate.

In these cases, it would seem that the ITS is most beneficial for users who use moderate amounts of HLH. It must be noted here that this is a correlation observation and not a reason for causation.

Difficulty levels of problems

Each problem in SQL-Tutor is assigned a difficulty level by the teacher. Difficulty levels range from 1 to 9 with 1 being the easiest and 9 the most difficult. There is a significant difficulty difference between each level. The Zone of Proximal Development (ZPD) [178] is the area just beyond a student's ability, or more specifically, it is the difference between what a learner can do without help and what they can just do, given some help by the tutor. In ITS terms, this is the zone where the student can just solve a problem with a little bit of help (i.e. LLH) from the ITS. This is also said to be where maximum learning occurs. Like most ITSs, SQL-Tutor attempts to keep the student within their ZPD by helping them maintain the correct difficulty level for their ability whilst guiding the student with increasing levels of LLH; the student is also free to use HLH as necessary.

We recorded the difficulty levels of problems attempted and problems solved and derived our own metrics: *maximum difficulty level of problems*

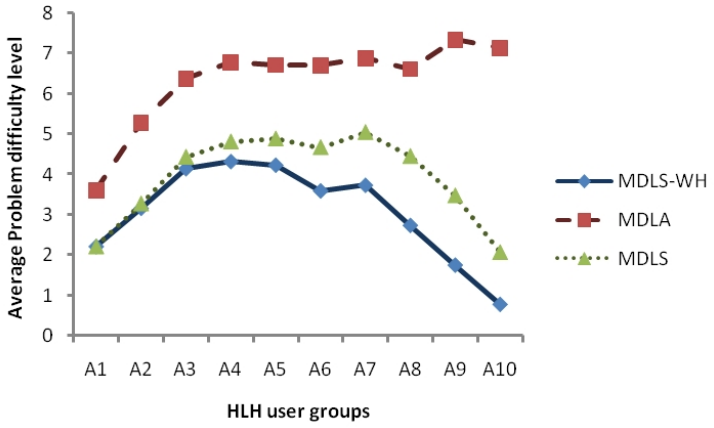

Figure 4.6: The MDLA, MDLS, MDSL-WH for all HLH user groups.

attempted (MDLA) and *maximum difficulty level of problems solved* (MDLS). MDLS simply gives the maximum difficulty of the problem solved without taking into account how much help the user sought in solving the problem, while MDLA simply gives the maximum difficulty level of all the problems that the student attempted. In our initial analyses we used MDLS to see how students from each group were able to learn the skills and progress through to solving more difficult problems. Manual analysis of the logs however, showed that in groups using high HLH, there was a large difference between difficulty of problems attempted and difficulty of problems solved; they attempted much more difficult problems than they solved. Further analysis of the logs showed that in groups that used higher HLH, there was a higher number of abandoned problems after at least one valid attempt was made.

Furthermore, in the higher HLH groups, there were large differences in solutions between the initial (incorrect solution) and the subsequent attempts as they got closer to the correct solution. It was as if the students in these groups were employing a "guess, check solution, then copy" strategy. In contrast, students in the lower HLH groups that started with an initial incorrect solution usually started with one that more closely resembled the structure of the correct solution, then made smaller changes as they got closer to the correct solution.

Due to this, we created another metric called *maximum difficulty level of problem solved without using HLH* (MDSL-WH). This is the maximum

difficulty of the problem a student solved, but without using any HLH. We determined that the MDLS-WH would give an indication of how far the user actually progressed through the domain on their own, and thus give us a clue, albeit a vague one, of the student's expertise in the domain.

The average MDLA, MDLS, and MDLS-WH for each HLH user group was calculated and is shown in Fig. 4.6.

As shown by the graph, the average MDLA is always higher than the the MDLS, irrespective of any help levels. This makes sense as students would attempt problems that were more difficult than their current capability, and end up solving problems in their ZPD.

The MDLS-WH increases steadily as HLH use increases, peaking at the $A4$ group (MDLS-WH = 4.32). After this, as HLH use increases, the MDLS-WH decreases quite quickly. The high HLH groups had very low MDLS-WH values (e.g. average MDLS-WH for A9 = 1.75 out of a possible difficulty level of 9). In fact, the lowest MDLS-WH was with group $A10$ at 0.77; this is interesting as difficulty levels in SQL-Tutor go from 1 to 9. In other words, students in this group solved problems without HLH that had a combined value less than the lowest difficulty value in SQL-Tutor. On further analysis, we discovered that this was due to several students not solving any problems on their own, without HLH. All these trends were also noticed in datasets B and C. Unlike dataset A, we had access to extra knowledge about the context (e.g. date of assignments, exams, etc) of the students in dataset B and C. In both sets, there was a spike of usage of the tutoring system closer to these prominent deadlines. This could provide us with one explanation as to students with their HLH ratio equal to or just less than 1. Students with very little time to study close to their deadlines might feel the need to satisfy their knowledge by moving through the questions quickly, simply viewing the full solution to the problems.

The lowest HLH user group ($A1$) also reported a low MDLS-WH average (2.2). This is similar to the trend found in the learning curves above.

What is perhaps most interesting about this graph is the difference between the MDLA and MDLS-WH graphs. On average, the higher the HLH usage, the greater the difference between the two graphs, with a particularly noticeable difference after group $A6$. Students who are low to moderate

users of HLH progress through the problem set, attempting and solving more problems on their own (without HLH). Students who are high users of HLH attempt problems way beyond their capabilities (in fact, they attempt the most difficult problems compared to the other groups) but solve only the easiest problems on their own. They also abandon more problems on average than other groups.

Although these techniques give us a way to mine data to find correlations, they do not give us reasons why these exist or what could be done to solve these problems (if, in fact they are problems). However, as educators, it has provided what we set out to do - give ourselves a tool to see if a potential pedagogical strategy could be built using data mining techniques and made ready for testing. Here, for example, the potential pedagogical strategy might try to limit HLH seeking behaviour to within the 0.4-0.6 (ideal mode) setting and see if that makes a difference to learning. This would open up a huge amount of research, such as "is this the ideal range for HLH?" or "are certain groups of people with higher meta-cognitive skills able to adjust their amount of help seeking to get the maximum benefit from their learning?" etc.

4.2 Exploring Help-Seeking behaviour in ITSs using EER-Tutor

Results from this study were published in [107].

We conducted the same analyses using a dataset from EER-Tutor. We will keep this section short as results found in this section were very similar to those found from the SQL-Tutor dataset. This section merely exists to confirm that these analyses were also conducted in EER-Tutor.

As mentioned earlier, EER-Tutor is an ITS that teaches conceptual database design using the Enhanced Entity-Relationship Model, and provides students with problems to practice their entity relationship modelling skills in a coached environment [123]. This tutor has also been used for many years and had several successful evaluations. The help levels in both tutors are the same making it easy for comparison. As with SQL-Tutor, we divided the domain help into LLH and HLH.

Although these two ITSs deal with database related areas, each domain is very different and requires very different skill sets from students. Further-

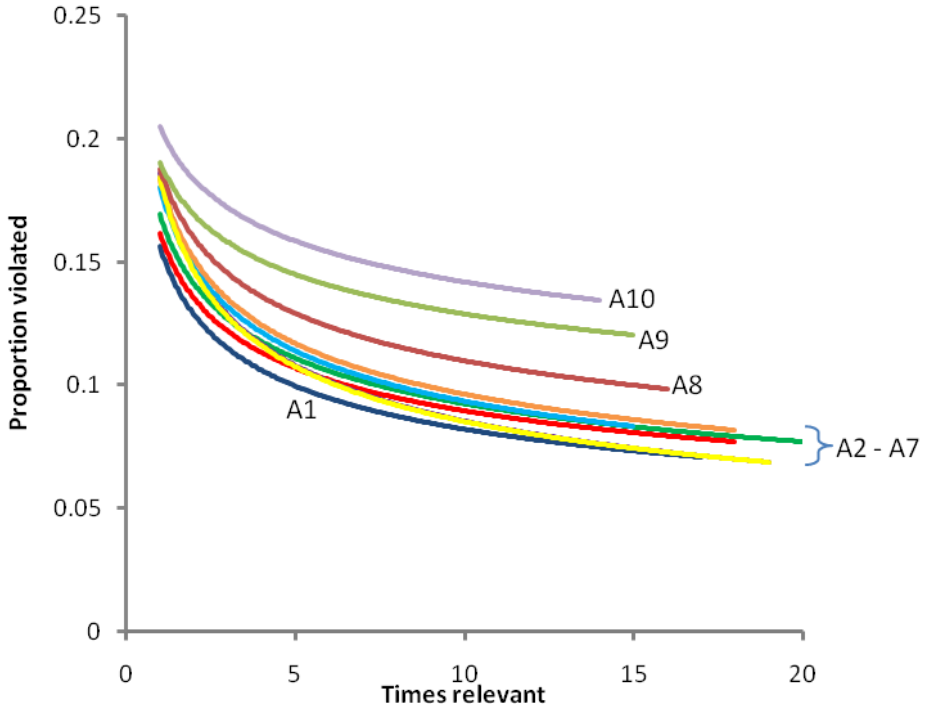

Figure 4.7: The learning curves for EER-Tutor

more, the method of solving problems (even to the point of textual versus diagrammatic) is considerably different.

As with the SQL-Tutor dataset, we excluded any student who had made fewer than 5 attempts. The final EER-Tutor dataset consisted of 936 students who made a total of 43,485 attempts, and spent just over 2,839 active hours on the system. The same analyses were conducted on this dataset, normalising the HLH useage, analysing the frequency graph, and examining the learning curves.

The learning curves were slightly different to the SQL-Tutor curves, but followed the same indications (see Fig 4.7). The power curve equations and fits are given in Table 4.2.

In this table, we can see that the moderate users of HLH did comparatively well again. The highest HLH users ($A9$ and $A10$) had the lowest power-to-curve fit (R^2), the lowest learning rate, and the highest error rate. These figures compare well with those from the SQL-Tutor datasets.

Table 4.2: Power curve equations and fits (R^2) for the ten HLH groups ($A1 - A10$) in EER-Tutor.

Group	HLH ratio	Power curve equation	R^2 fit
A1	0.0 - 0.1	$y = 0.156x^{-0.28}$	0.959
A2	0.1 - 0.2	$y = 0.162x^{-0.25}$	0.963
A3	0.2 - 0.3	$y = 0.169x^{-0.26}$	0.963
A4	0.3 - 0.4	$y = 0.185x^{-0.33}$	0.938
A5	0.4 - 0.5	$y = 0.181x^{-0.28}$	0.967
A6	0.5 - 0.6	$y = 0.183x^{-0.28}$	0.947
A7	0.6 - 0.7	$y = 0.184x^{-0.33}$	0.978
A8	0.7 - 0.8	$y = 0.189x^{-0.23}$	0.954
A9	0.8 - 0.9	$y = 0.190x^{-0.16}$	0.858
A10	0.9 - 1.0	$y = 0.204x^{-0.15}$	0.878

4.3 Implementing a pedagogical strategy in an ITS

In the previous section, we found that using data mining techniques, educators could find the harvesting ground for potential pedagogical strategies. In this section, we want to see if this could be implemented in a simple prototype so that it tries to follow our original design of separating the Pedagogical Control Centre (PCC) and the Pedagogical Strategy Set (PSS) from the main ITS, hopefully making it more flexible (more strategies can be added), easier to use (one does not have to be an ITS expert to use it), and reusable (the idea could be used for more than one tutor).

The prototype we implemented was quite simple. It had three modules in addition to the standard ITS; see Fig. 4.8. The Pedagogical Student Model (PSM) was a student model that showed how this particular strategy was used by each student. It is envisioned that each pedagogical strategy would use at least one PSM per student. In our case, it consisted of the history of using HLH for each attempt in the form of a list. If the student used HLH, the list was appended with a 1 or in the opposite case, a 0. After a while, the PSM built up with a history that looked like (0 0 1 0 0 1 1 1). This means that four of the eight attempts were using HLH; the latest three used HLH.

The PSS in our case only dealt with one pedagogical strategy: keeping

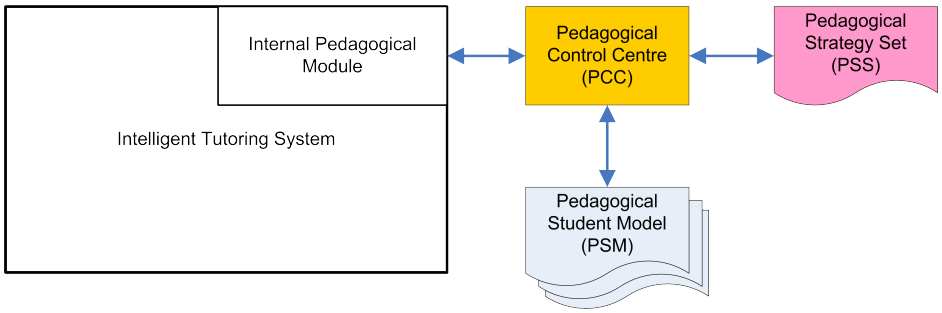

Figure 4.8: The initial design of the framework, including the PCC, PSM, and the PSS.

the student between the ideal 0.4 to 0.6 HLH usage. We got these numbers from the datamining process we conducted earlier where this range seemed to be the ideal help seeking setting for effective learning. In this particular case we would want to study whether this was a causation effect (as opposed to the correlation effect found earlier). The strategies were implemented using constraints. Here are a few written in pseudo code below. Each constraint gives a specialised case giving us the ability to also give specialised feedback. The teacher could write as few or as many constraints as they need; the specialisation of feedback given would guide this decision. In our case, the strategy was employed as soon as the student did more than 5 attempts for any given problem; this could be adjusted by the educator as deemed fit.

Constraint example 1

Relevance condition If number of attempts > 5 then
Satisfaction condition HLH must be between 0.4 and 0.6 (inclusive).
Feedback on violation Your use of feedback is not ideal. Ideally you would be mostly using Low-Level Help (up to hint) and some usage of High Level help (above hint) when you get stuck. You can change the feedback levels using the feedback level combo box.

Constraint example 2

Relevance condition If number of attempts > 5 then
Satisfaction condition HLH must be ≥ 0.4

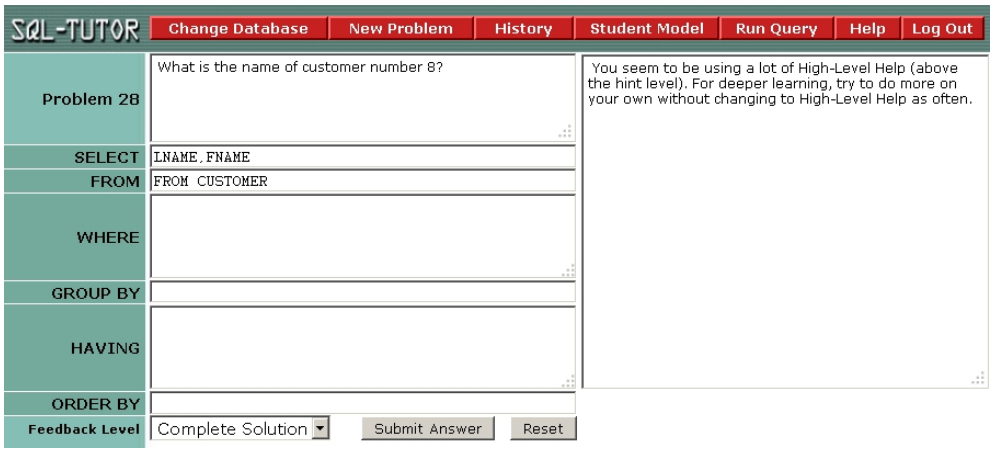

Figure 4.9: SQL-Tutor showing feedback on violation of one of the pedagogical tactics.

Feedback on violation Are you using enough help? Remember you can use the feedback level combo box to get more help if you are stuck.

Constraint example 3

Relevance condition If number of attempts > 5 then

Satisfaction condition HLH must be ≤ 0.6

Feedback on violation You seem to be using a lot of High-Level Help (above the hint level). For deeper learning, try to do more on your own without changing to High-Level Help as often.

The PCC is where the logic resides. The PCC communicated with all modules: the pedagogical module in the ITS, the PSS, and the PSM. It knew about all the strategies available (in our simple case, only one) and about all the types of PSMs available (in our case again, only one per student). It knew what information it needed to parse each attempt and requested this from the ITS. After using information from the ITS and the PSM, it used the *constraints* from the PSS to parse the attempt and provide the ITS with appropriate feedback. See Fig. 4.9 for a screenshot when one of the pedagogical tactics was violated.

4.3.1 Lessons learnt

We never got to test this prototype in a real classroom situation. It was used in the research lab where portions of actual actions of students (taken from their logs) were passed through this system to test that it worked.

- In this small context with a simple pedagogical strategy, it worked. It monitored and instructed the student to keep within the 0.4–0.6 margin. Remember that within the context of this project, we are not trying to determine the correct margin or whether this pedagogical strategy is even valid. We are instead trying to see if we could design a framework where educators could try their own pedagogical strategies easily. Within that context, this was at least a partial success.

- This was used as a proof of concept to see if we could even use such a design. The answer is definitely a "yes". However, as more pedagogical strategies get implemented, the system would become more complex. In certain cases, some *parts* of the strategy could be used as parts in another strategy. For example, keeping the student's usage of HLH low might be used in this strategy as well as another that aims to prevent gaming, or one that is aimed at top students to push them a little harder than the rest of the class. This means that a strategy is not atomic and is different from the components that make up that strategy. The strategy contains the overall goal (what the educator wants to achieve) while the smaller components (which we called *pedagogical tactics*) work together to provide the implementation to achieve that goal. These tactics could be reused in certain cases to fulfil the goals of other pedagogical strategies.

- Our simple strategy kept the student within that margin when in fact there were cases when the student did not necessarily need to be within that margin. One example was of a student who was doing quite well on their own without the need for HLH, however, our system urged them to use more. These are for example, students from the $A1_1$ group. Again we believe that this is for educators to choose how they want

their pedagogical strategies implemented. For example, it would be a simple case for the PCC to check with the ITS and see how well the student is doing. There would be a tactic within the PSS that allows a good student to carry on without forcing more help on them i.e. group $A1_1$ would be allowed to have a low HLH ratio while the margin would be enforced on group $A1_2$.

- We tried two simple algorithms in the PCC to get data from the PSM. The question was "how much data from the student's PSM should be used to make a decision on using a tactic?". In our first algorithm, we used all the data from the PSM. Although this gave quite a good indication of the student's overall usage, it was very slow to react to change; the bigger the PSM, the slower it reacted to change. For the second trial, we used a sliding window algorithm. This meant that we only looked at the student's most recent attempts and used that to calculate whether they were within the margin. This reacted to change very quickly and was able to advise the student straight away. However, this also caused, what we called, strategy thrashing, because it changed so quickly. As soon as the student went out of the required margin (within the sliding window), the strategy changed. It only then took a little while for the student to get back into the margin, wherewith the strategy changed again. We believe that if one were to use this strategy, it would be necessary to spend some time to come up with a better size for the sliding window (i.e. how recent is recent?) so that it was relatively quick to react, but not so quick that it caused strategy thrashing. This also depends on the consequenses or measures the ITS takes on violation of a pedagogical tactic (or how quickly a student reacts to the feedback).

- In our current system, when a pedagogical tactic was violated, the PSS simply passed on the feedback message via the PCC to the ITS. The ITS then had full control on what to do with that message. However, there are many more exciting possibilities. The PCC has all the history available in terms of pedagogical tactics. This means that it can gauge

a severity measure. This measure would be based on the educator's decision. For example, a violation of a tactic might be more severe if a student continually violates it. Or, it might be considered more severe the further away the student is from the margins set for *ideal* student behaviour. The PCC could send this information (say, as severity levels) to the ITS. The ITS could then make a decision on what to do. For example, for low severity levels it could simply prompt the student with a feedback message, whereas for higher severity levels it could enforce stricter guidance (e.g. in our case, not allowing any changes to the feedback level combo box). These measures however, call for either greater coupling between the modules (e.g. the PCC and the ITS) or for some form of standardisation.

- We used a domain-independent strategy for our trial. This meant that there was much less coupling between the modules. However, domain-dependent strategies might have to either increase the coupling between the PCC and the ITS or perhaps a better way is to still have a pedagogical module (albeit smaller) within the ITS that deals with domain-dependent strategies. This would make the PCC/PSS/PSM more easily able to be used with other ITSs.

Chapter V

Understanding Constraints in terms of Learning

While our potential framework design does not specify any particular type of ITS or knowledge base, we decided to use Constraint-Based tutors and to model the PSS as constraints, mainly because we had access to a number of mature and well-evaluated CBM ITSs. Even though this framework does not call on any particular type of domain modelling to be used (as it does not rely on any specific part of the way the domain is modelled), we have included this Chapter to give an idea of the method we have chosen: CBM [136].

Knowledge is often divided into two separate categories: declarative and procedural [138]. Declarative knowledge is knowledge about facts while procedural knowledge is knowledge about how to conduct a procedure (i.e. a skill). Declarative knowledge can be learnt via a third party, such as a book, the internet, or a teacher. Procedural knowledge can only be learnt by proceduralising the declarative knowledge; this is accomplished by engaging in deliberate practice. As an example, simply reading and memorising a book on motor mechanics might mean that one has good declarative knowledge about the subject. However, that does not mean that they would necessarily be able to apply that skill. To master working on engines, one must practice that knowledge and convert it into procedural knowledge. There are several examples of skill based knowledge requiring proceduralisation. Learning a recipe by reading it is one thing; actually cooking the dish is another.

Cognitively, when a person has a final goal situation in mind, they use the *perceive-decide-act* cycle [137] within the current situation to move to a situation that is closer to that final goal, i.e. the person decides on an intermediate goal, gets information (through perception) from within their current situational environment, decides on what must be done, and then acts

on it to get to the next situation. Latencies in behaviour between domain experts and novices can be explained using this cycle; novices take much longer at each stage, especially as they contemplate what they perceive in their current situation with the myriad of actions they could perform. The action they take is not independent or modular, but is dependent on the goal and situation. This part of the cycle (of a goal in a particular situation requiring a particular action) is called an individual production rule; each production rule is modular. A production rule is the relationship between a goal, situation and action, (see Equation (5.1)) given that the action is to take a person from the current situation towards the goal. Cognitive Tutors aim to guide a student from an initial problem situation to a final goal, over many steps, using a number of production rules. They concentrate on giving guidance on the individual actions taken by a student in a situation to get to a particular goal. Additional feedback on incorrect steps could be given by modelling incorrect production rules (also known as the bug library) [174, 118].

$$Situation, Goal \rightarrow Action \tag{5.1}$$

As students learn, they are creating new rules, specialising overly generalised rules, and either increasing or decreasing the strength of each rule depending on the outcomes from using that rule. The strength gives a probability of that rule being used again in a similar situation with a similar goal. The current stock of rules held cognitively by a person is the sum of their current practical knowledge.

5.1 Learning from performance errors

Ohlsson, in his paper *Learning from performance errors* [137] describes a different method of learning; the idea of learning from the errors one makes. In his theory, he sees errors as opportunities for learning. But for learning to take place, two things must occur: 1) the student must recognise that they are in an error state, and 2) the student must figure out how to correct their error and get to a correct state. Unlike with production rules, this theory concentrates on correct/incorrect states rather than correct/incorrect

actions. An error thus just becomes a longer way to the correct solution.

There are generally three accepted ways by which an error can be detected: recognising it by oneself, expectations versus reality, and being told by a third party. *Recognising it by oneself* is when straight after a mistake has been made, the student realises and catches themselves making an error; this in many cases is called a slip. To catch themselves making the error requires the student to have good declarative knowledge. For example, after having learnt the steps to the Tango, a person has the ability to instinctively catch themselves placing themselves into an incorrect stance. *Expectations versus reality* is a situation where the student expected one outcome, but instead got another. They have detected that something must have gone wrong as reality does not match their expectations. While dancing the Tango, the student (a man), realises that something has gone terribly wrong when he hears shrieks of pain from the lady and realises he has mis-stepped and just put his entire body weight on her foot. This reality (the shrieks of pain) was far from his expectations; he has detected an error. The third way of detection is for *a third party (like an instructor) to inform you that you have made an error*. The dance coach, while watching the Tango dance duo can take instructive steps when he/she detects errors. The errors might have otherwise gone unnoticed by the students. Often the ITS takes this approach and acts like an instructor helping you detect that errors have been made.

Figuring out the error and getting to the correct state requires the student to gain declarative knowledge, preferably with guidance. This is where the ITS acts as a guide again, and provides the missing declarative knowledge and slowly guides the student to the correct state.

This is where this theory (Constraint-Based Modelling) differs significantly from Model Tracing used in Cognitive Tutors. Instead of focusing on the incorrect actions that brought the student to the incorrect situation and trying to correct it by finding correct actions, CBM focuses on the current state and on correct knowledge only. It then helps the student detect any errors (comparing it to correct declarative knowledge) and then guides them to a correct state (by teaching them correct declarative knowledge).

5.2 What are constraints?

The knowledge base (domain knowledge) in a Constraint-Based Tutor is made up of a set of constraints. This set could be quite large; SQL-Tutor has over 700 constraints. Each constraint is an atomic fact or principle of the domain. Each constraint is in the form $< C_r, C_s >$, where C_r is the relevance condition and C_s is the satisfaction condition. C_r specifies when this constraint is relevant in the student's solution. If this constraint is relevant, then C_s must also be true in the student's solution. If C_s is not true in the student's solution when it is relevant, then the constraint has been violated and an error has been detected. A constraint only refers to correct domain knowledge. Thus the set of constraints has the ability to identify correct solutions from the space of all solutions. Quite often, one or more feedback messages are also attached to each constraint, which could be displayed to the student on constraint violation; each feedback message could give further help if the same constraint is violated repeatedly.

Each constraint follows a similar pattern with the relevance and satisfaction conditions being similar in form to an "if/then" statement:

Relevance condition IF something in the student's solution matches the relevance condition,

Satisfaction condition THEN it better be that the satisfaction condition also be true (i.e. the satisfaction condition given here must match in the student's solution).

Feedback message on violation Otherwise give them this feedback.

As an example, a cooking simulator ITS might have the following constraint (in pseudo code):

Relevance condition IF the chicken has been taken out of the oven, ready to be served

Satisfaction condition THEN it better be that the chicken is cooked thoroughly.

Feedback message on violation "The chicken is not cooked thoroughly. You must not serve uncooked chicken."

The specificity of the feedback message is proportional to the granularity of the constraint; the more granular the constraint, the more specific the feedback message would be. This allows the author to choose the granularity at which the constraints are written; ideally a constraint is an atomic part of the domain.

The student model in a CBM tutor contains the history of each constraint, preceded by its identifier. (164 (1 0 0 1 1 1)) means that constraint 164 was relevant six times. Out of those six times, it was violated twice. A constraint is only modelled as being relevant on student solution submissions; simply viewing the question does not change the student model. As stated earlier, a single attempt in tutors like SQL-Tutor and EER-Tutor could contain multiple steps or even the whole solution. Thus, each attempt could have multiple constraints relevant and many could be either satisfied or violated at once. In this case, feedback needs to be chosen wisely, so as not to overwhelm the student.

Our question in this part of the research (as we are using CBM tutors) is: "Is there any correlation between seeing (experiencing) a constraint and learning?".

5.3 Do students who see more concepts in an ITS learn more?

The following section has also been reported in [102].

In the following sections, when we say that a student *sees* a constraint, we mean that the student attempted and submitted a problem where this constraint was relevant.

Researchers for many years have been telling us the importance of active participation by students in a learning environment [142, 5]. Active participation means truly connecting with the material, engaging with it, using exploratory skills to find out more than just the basics, and interacting with it through problem-solving [144].

It is easy to see and understand this in our human tutoring situations. In traditional lectures, this could be a problem as lecturers simply become the givers of knowledge while students are the recipients. A student could see several concepts in class and not interact with any and thus have low

amounts of learning. An example is that of a student who spends most of their time daydreaming in class while the lecturer still keeps introducing them to new concepts. However, in a one-to-one teacher-to-student ratio (say private tutoring), it is much harder for the student to mindlessly experience the new concepts. A good human tutor would engage the student in exercises, concentrate on correct knowledge, guide them through the more frustratingly difficult parts of the problem, and give feedback as necessary.

How does this work in ITSs though? If a student sees (experiences) more constraints, do they necessarily learn more? Does this same basic principle apply in ITSs? Can students use an ITS and learn nothing because they are so passive with their learning?

Basic observation shows differences between the way a student learns in a traditional lecture environment and while working on an ITS. In a traditional lecture environment, the material (type and difficulty) and rate of moving through the material is provided by an external source, the lecturer. There is very little input from the student. However, in an ITS, the student must initiate all progress, i.e. movement through the material is solely based on the student initiating the next move. This might be as small as selecting the next problem or requesting help. As such, the student cannot sleep through a session in an ITS. Furthermore, good ITSs are based on the student doing something and the ITS providing some form of feedback (not necessarily in textual form). This feedback is tailored to each student and to what they just did. Many ITSs also use the student's historical model while making decisions, such as which problem to suggest to the student to do next (i.e. it is customised to that student). For a student to disengage totally from learning would mean not progressing through the ITS. There is usually no external entity built into the ITS that controls and maintains the flow and delivery of material at a particular rate. Progress is student dependent. Due to this basic difference, one would assume that if a student progresses through problems within an ITS, they must be involved in some way to the material they are interacting with. As active participation is correlated to learning, we could assume that students who progress through an ITS must learn something, no matter how small it might be.

To see if this assumption holds, we mined and did analyses on data from

an online version of SQL-Tutor. SQL-Tutor has many problems varying in difficulty level from 1 to 9 with 1 being the easiest and a significant difference in difficulty between the levels. Each problem might have multiple correct solutions. Each solution might have many constraints that are relevant (that need to be satisfied in order for the solution to be correct). The number of constraints per problem and/or the complexity of those constraints influence the problem difficulty level. This means that when solving more difficult problems, the student could be dealing with either a large number of constraints and/or more complex constraints.

5.3.1 The data and the methodology

The data (student models and logs) for these analyses were collated from an online version of SQL-Tutor. Students that made fewer than five attempts were excluded from these analyses. Remember that one attempt does not necessarily mean one step; it could be a full solution to the problem. The final dataset consisted of 1,803 students who spent just over 1,959 combined hours, while making a total of 104,062 submissions. A student might have made several submissions while solving each problem. In total, these students solved just over 70% (19,604) of the problems they attempted (27,676).

There are certain basic constraints that are relevant to all queries (problems). These are principles that are very easily learnt and do not provide in-depth knowledge to the domain; these were therefore excluded from the study. The number of constraints that students saw at least once while solving problems varied from 6 to 333. While engaged in their problem solving activities, students saw (experienced) a combined total of 174,309 constraints.

The methodology used was simple. To calculate *constraints seen*, we counted all the unique constraints in each student's model. To calculate *constraints learnt*, we used two methods, both of which gave very similar overall values in the end. In both cases, we used a window to capture the students latest attempts at a constraint. In the first method, the window size was 5 if the student made 5 or more attempts at that constraint; otherwise the window size was 3. In the second method, the window size was always set at 5. This meant that the *constraint learnt* ranged from zero (not knowing

the constraint) to one (knowing it well).

Here are a few examples of calculating the "constraint learnt" value using the first method:

Constraint history (0 0 0 1 0 1 1 1 0)
Which window size? As there are more than 5 attempts, we will use the window size = 5 method.
Constraint learnt In the last five attempts, there are 3 correct attempts and 2 incorrect attempts. Therefore, constraint learnt = $\frac{3}{5}$.

Constraint history (1 1 0 1)
Which window size? As there are fewer than 5 attempts, we will use the window size = 3 method.
Constraint learnt In the last three attempts, there are 2 correct attempts and 1 incorrect attempt. Therefore, constraint learnt = $\frac{2}{3}$.

Here are a few examples of calculating the "constraints learnt" value using the second method:

Constraint history (0 0 0 1 0 1 1 1 0)
Which window size? Always 5.
Constraint learnt In the last five attempts, there are 3 correct attempts and 2 incorrect attempts. Therefore, constraint learnt = $\frac{3}{5}$.

Constraint history (1 1 0 1)
Which window size? Always 5.
Constraints learnt Constraints learnt = $\frac{3}{5}$.

We did this for each unique constraint for each student. Each method makes an assumption of how many attempts are required to determine the state of learning for each constraint. We used these methods as they were used somewhat successfully in various versions and evaluation studies previously.

We calculated the average difficulty level for problems attempted and problems solved for each student. This is shown in Fig. 5.2 and Fig. 5.3.

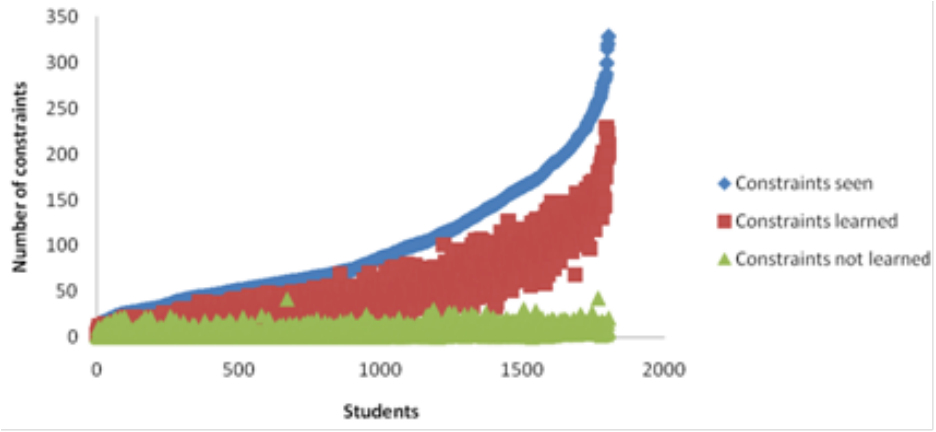

Figure 5.1: Constraints seen, constraints learnt, and constraints not learnt for each student, ordered by constraints seen.

5.3.2 Results and discussion

Constraints

As seen in Fig. 5.1 and from the correlation calculations, the number of constraints a student sees is strongly correlated to the contraints learnt (Pearson's $r = 0.947$). This means that students who saw more constraints overall learnt more. Remember that to have seen a constraint, they must have made at least one attempt where that constraint was relevant. That also means that they would have got some feedback if that constraint was violated. Simply moving through and viewing the problems (without doing anything) is not counted in this case. The variability in the constraints learnt could be due to individual differences, quality and quantity of active engagement, use and amount of help, gaming [21], and low-level skills (e.g. low metacognitive abilities). This supports our original hypothesis that learning on an ITS requires some form of engagement which translates to learning.

The number of constraints not learnt remains relatively constant for all users, regardless of the number of constraints seen. This makes sense, as the proportion of constraints that are wrong decreases as the number of constraints seen increases. However, what is interesting is that if we look at Fig. 5.2 and Fig. 5.3, we can see that students who saw more constraints (to the right of the graphs) on average also attempted and solved more difficult

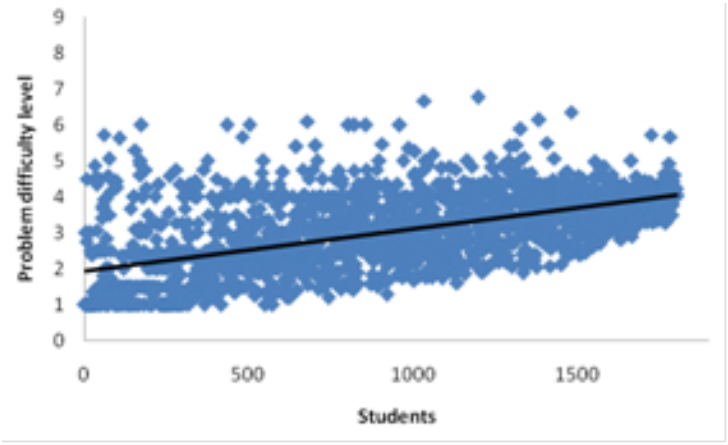

Figure 5.2: Average difficulty levels of problems attempted for all students, ordered by constraints seen.

problems (i.e. saw more complex constraints) than those who saw fewer constraints (to the left of the graphs).

This supports the hypothesis that students who see more constraints not only learn more, but progress on to more difficult concepts.

Time

Fig. 5.4 shows the number of constraints seen, plotted against the total time spent in the system. From the graph we can see that total time spent is proportional to number of constraints seen, i.e. the students who saw more constraints spent longer in the system. This is expected as the greater the number of concepts explored in the system, the longer the student would take in the system.

Fig. 5.5 and Fig. 5.6 show the time spent per constraint seen and learnt by each student. In both these figures, students who have seen fewer constraints are to the left of the graphs. This means that even for relatively difficult constraints, students on average still spent the same amount of time on each constraint. This could be because students who have more expertise are the ones progressing onto seeing more constraints. The outliers could be students who are attempting more difficult problems than their expertise level.

In conclusion, on average:

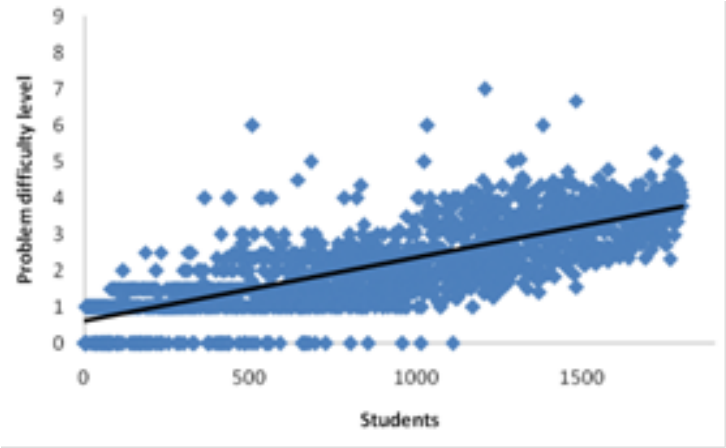

Figure 5.3: Average difficulty level of problems solved for all students, ordered by constraints seen.

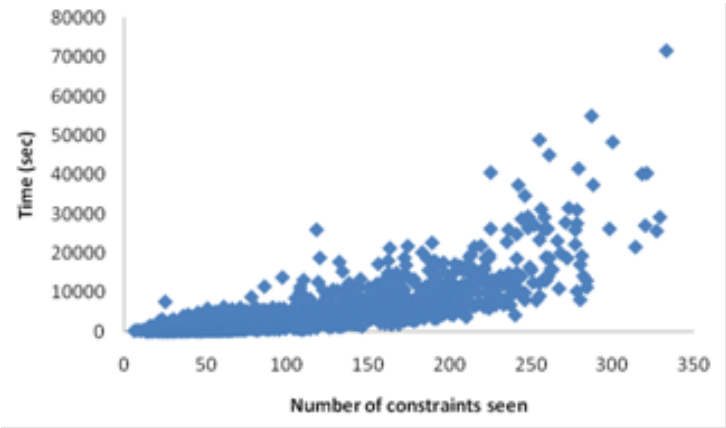

Figure 5.4: Number of constraints seen against total time taken by students.

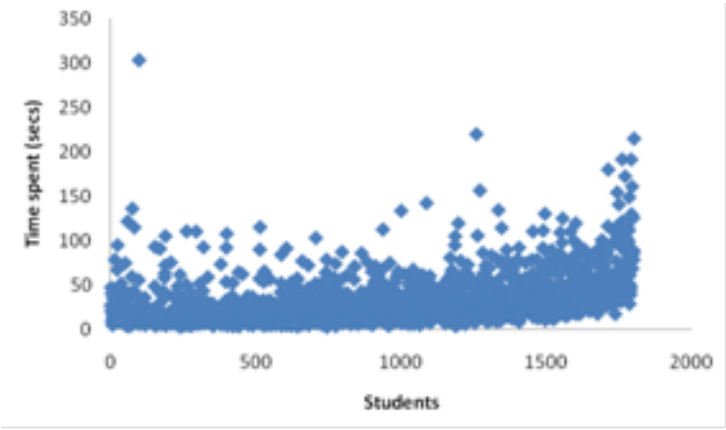

Figure 5.5: Time spent per constraint seen for each student.

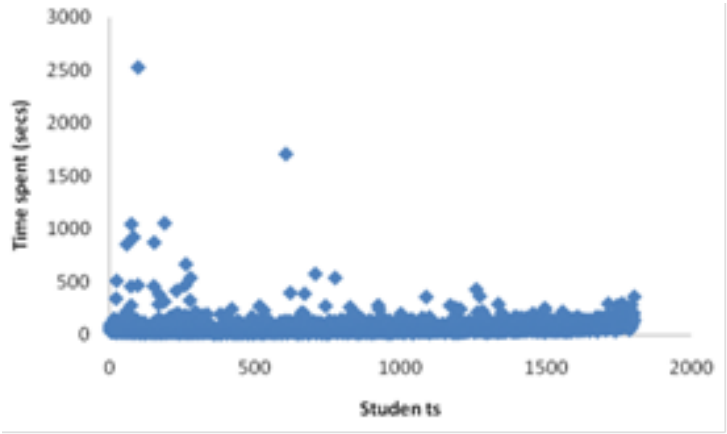

Figure 5.6: Time spent per constraint learnt for each student.

- students who see more constraints learn more.
- students who see more constraints proceed onto more difficult problems that contain either more constraints or more complex constraints.
- Although students who see more constraints spend more time in the ITS, the time per constraint remains the same for all students, i.e. as students' expertise increases, they do not take longer on the more difficult constraints.
- Individual differences exist, possibly due to quantity and quality of time spent, metacognitive abilities, and amount of help (particularly HLH) used.

Chapter VI

Validating a strategy by using a human tutor to control certain aspects of a complex strategy

The results of the study described in this Chapter were published in [104].

Quite often, prior to implementing a complex intelligent module, ITS developers substitute that module with a human tutor to validate the design before investing many hours of work actually implementing it. One such example is a Wizard of Oz study. Here a human tutor takes the place of a computer, and through some pre-written scripting process communicates with students; the students are unaware that they are actually communicating with a human. In this manner, a lot of data can be gathered (including the feasibility of the design) that could help with designing and implementing the real solution.

After our success with the design of our framework using a simplistic (help-seeking) strategy, we decided to revisit the design, but this time with a more complex strategy (that has not been implemented in an ITS yet) in mind. Would our design hold up to more complex strategies or do alterations have to be made to the design? To try this, we took the approach of using a human tutor first to simulate some parts of our Framework in order to validate the strategy before proceeding with the implementation.

6.1 Framing a Problem-Solving Scenario in an ITS

Unfortunately, using any strategy in a classroom situation is very difficult to evaluate. ITSs provide an objective method of providing such evaluations. We believe that, if our framework design is feasible, adding strategies and evaluating them objectively with ITSs under different conditions, would add significantly to the knowledge of the teaching and learning domain. Until

ITSs were introduced in schools, many of the educational evaluation studies were based on cases or in single classroom studies, without many controls. Having a framework where teachers could one day relatively easily add their own strategies and have their own ITSs change behaviour accordingly would mean a big change from the way we look at teaching strategies currently.

Framing a learning scenario (shortened to "Framing") is a strategy that has been used by teachers in classrooms with varied results. In fact, it is a complex strategy; one that is made up of other strategies. It is prescribed by educational facilities, and teachers are encouraged to use it as it seems to have all the right ingredients that potentially help maintain high student engagement levels. However, there is no research directly related to Framing. A software that could potentially allow educators to place such activities in sequence and allow for Framing is the Learning Activity Management System (LAMS).

LAMS is a software package in which educators can enter in their own learning designs into LMS type learning environments. Learning designs (LDs) are sequences of learning activities which involve groups of learners interacting within a structured set of collaborative environments. Learning designs focus on the dimensions of content context (rather than just content alone), are more activity based (rather than absorption-based) and place a greater role on multi-learner (rather than single-learner) environments [52]. LDs also emphasise adaptation. As the content is secondary to the LD itself, the content could simply be replaced and a new LD for a new discipline could be ready in a short amount of time. LDs are also reusable and pedagogically neutral. Being pedagogically neutral means that they do not subscribe to any particular pedagogical approach or theory, but that the educator can specify any learning activities as they desire [53]. The original LAMS software was built on EML (Educational Modeling Language) and a form of IMS-LD. However, in the new version, a detailed "tools contract" specifies the requirements for all activity tools that integrate with the new LAMS V2.0 "controller". There are now interfaces with the authoring, monitoring, learner and administration environments of LAMS, including technical details for tool deployment [54].

A learning session is *framed* when the learning activity is immediately

preceded by a pre-action (or *priming*) phase and followed by a post-action (or *reflection*) phase. All parts of the framed lesson occur in sequence during the same session. In our case, the learning scenario was solving problems within an ITS.

Framing assumes that students have some knowledge about the domain; for example, they have at least attended relevant lectures prior to the framed session. During the pre-action phase the teacher reintroduces the target concepts (concepts that would be used during the problem-solving phase), links them to previously learnt concepts (explains where these concepts fit into the domain), works through examples, and discusses the common misconceptions that students have in this area. S/he also sets the boundaries to this particular learning scenario. The pre-action phase is not similar to a lecture in either its intended purpose or necessarily in its content. In the pre-action phase, students are not merely recipients of knowledge. They are engaging their brain in thinking and focusing on the crucial topics that they will need to know for the learning phase.

In the problem-solving (or learning) phase, students solve problems related directly to the target concepts that were defined by the boundary in the pre-action phase. This could either be an individual or group activity. The teacher acts as a guide during this phase.

In the post-action phase, the teacher prompts each student to reflect on his or her (positive and negative) problem-solving experiences. Students are encouraged to analyse their errors, including the source of these errors, thereby uncovering their misconceptions. The common mistakes made during the problem-solving session are usually worked through as a group. The teacher is actively involved in the pre-action and post-action phases. They may act as a guide during the problem-solving phase.

There are several theories that make this a plausible strategy. Unlike in a traditional lecture, the pre-action and post-action phases are both very interactive. Even though the teacher directs these phases (sometimes quite strictly), it relies on a lot of input from the students. Learning that is active and not passive, leads to higher learning gains [43].

Although our long-term memory is potentially limitless in size, we have limits on our working memory; this basically defines how much we can deal

with at any given time. Experts, over time, have developed strategies to organise information into schemas that allow them to pull vast amounts of related knowledge from their long-term memory, helping them cope with difficult and complex situations in spite of working memory limitations [60, 101]. Cognitive Load Theory [91] suggests that problem solving for novices generates heavy working memory loads, which could be detrimental for learning. When a student is presented with new items to learn, they have not yet organised and added this to their long-term memory schema. Each item takes up the space of one separate item in working memory, bringing the working memory limit into play very quickly. To balance this load, just prior to the learning session, teachers should help narrow the problem search space by creating "boundaries" to each session, and alleviate the working memory restriction by making sure that only items relevant to the task are loaded into working memory.

Meta communication about the presentation of the subject is important with respect to learning [99]. Prior to the learning activity, the student needs to know what concepts the learning activity will contain (the boundaries of the learning segment), which should be well-defined by the teacher. The student needs to know the content of the session, differentiating between the old and new material. Teachers should help the student see where these new items are positioned within the schema of the domain - so as to help the student create these new links between existing and new knowledge. Students need help to create the links between the new knowledge and previously learnt knowledge [99]. This is exactly what happens during the pre-action (priming) phase.

Learning models suggested by educators certainly have many differences between them; some more subtle than others. However, for our case, many learning models view learning as a cyclic process, around which knowledge acquisition, knowledge application (including experimentation and experience), and reflection (or knowledge processing) occur. Andreasen and Wu [15] discuss a few of the commonly used experiential models. Framing is a simplified (and thus possibly, easier-to-implement) version of many of the models.

Reflection promotes deep learning [86, 85, 98]. Critically analysing the

experience after it has occurred helps challenge the student's underlying perception of the domain or topic, identify and correct misconceptions, and integrate new knowledge with existing knowledge [17]. This also helps in transferability, as the student slowly becomes able to transfer the new knowledge to other types of problems or scenarios. Self-explaining one's actions [47] and monitoring one's progress via open student models [87, 119] have been shown to be useful reflective tools that benefit learning. Reflection helps students discover if their previously held knowledge still holds true.

6.2 Design and methodology

We chose *Framing the lesson scenario* as our teaching strategy because of many reasons. Three of the main reasons were that: 1) it added quite a bit more complexity than our Help-Seeking strategy, 2) Some aspects of Framing could be easily conducted by an expert tutor who knew both the domain and the internal pedagogical structure of the ITS without compromising data collection while we gathered data about its validity, and 3) Framing happens at certain points of the session (beginning and end) rather than right throughout the learning session, making it easier to use the human tutor with multiple participants. The human tutor took the roles of both the PCC and the PSS (see Fig. 6.1) and conducted both the pre-action and post-action phases. In this situation, the strategy lies fairly dormant during the problem-solving phase (i.e. it does not try to influence the student during the problem-solving phase), we thought that the tutoring modules did not have to keep track of the strategy useage by having actual PSMs. Instead, as the teacher interacted with the students in the post-action phase, the students' mental PSM would be shown to the teacher, who could then address any issues that occurred during the problem-solving phase.

The study was held at the University of Canterbury in 2008. The ITS chosen was SQL-Tutor. SQL-Tutor is usually used in a second-year paper (COSC226) at the University, that teaches database concepts. COSC226 students are expected to attend lectures and a two-hour lab session (where they can practice what they learnt during the lectures) each week.

Thirty eight students who volunteered from COSC226 participated in the

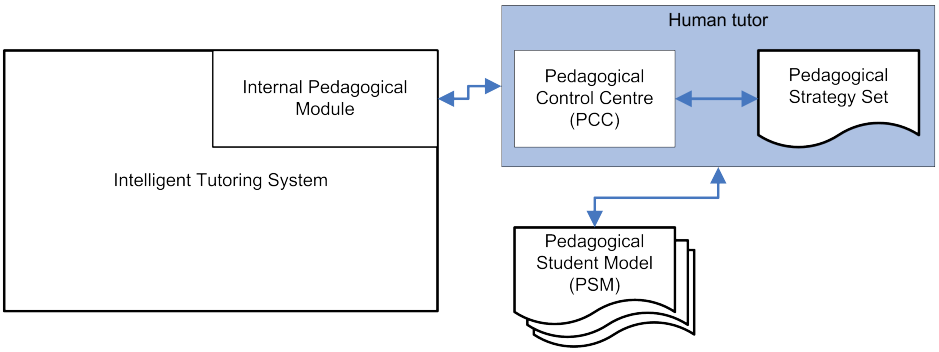

Figure 6.1: Using a human tutor to simulate parts of the module

study for no monetary reward. They were divided randomly into two groups: experimental and control. The control group just used SQL-Tutor during the session, while the experimental group also had the Framing condition (see Fig. 6.2). None of the participants had used SQL-Tutor previously, although they all had experience with another ITS (EER-Tutor) during the few weeks prior. We wanted to perform the evaluation in a setting that was as close to the normal teaching environment faced by students in COSC226 and similar courses. As such, the experimental and control sessions were held during regular course lab sessions (100 minutes long) on 14 and 15 May 2008 respectively. The students participated in the study during the lab session they normally attended throughout the course. The lab sessions are usually held after they have covered the material in the lectures (both groups attended the same lecture). Both groups were informed that their performance during these sessions did not count towards their course grade.

Framing includes setting up boundaries (containing target concepts) for the problem-solving session and going through these target concepts in the pre-action phase. A set of target SQL concepts, namely the concepts covered by SQL queries using the GROUP BY and HAVING clauses were chosen for the study. These types of queries involve concepts that students generally find difficult as they add another level of complexity to the *Select* statement. With the GROUP BY clause, students not only have to deal with tuples (rows), but also in terms of grouped tuples. Furthermore, conditions on tuples can be set in the WHERE clause, but conditions on groups of tuples have to be set using the HAVING clause, which is another common point of

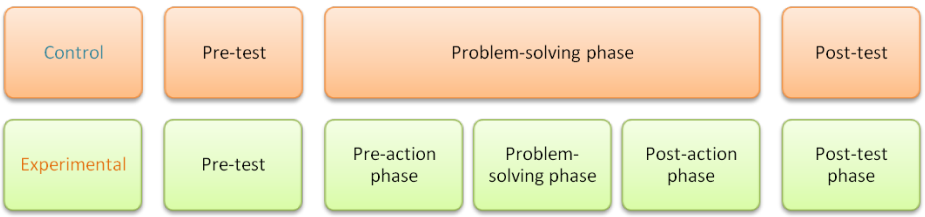

Figure 6.2: The phases in the control and experimental versions

confusion amongst students. Using groups of tuples now allows students to use aggregate functions (functions on the whole group, such as *sum, average, count*, etc.). This adds further complexity. For the problem-solving phase, SQL-Tutor was restricted to only present problems relating to these target concepts.

The experimental and control sessions had five and three phases respectively within the set 100 minutes; this is shown in Fig. 6.2. Initially, students in both groups completed an identical pre-test in the first phase and an identical post-test in the final phase. Both tests were different but comparable to each other. Each test contained three questions that related to the target concepts. For each question, students were required to formulate an SQL query. A domain expert created and marked all the tests. The domain expert was blind to the group of each test. The three questions varied in difficulty and marks were given for correct usage of SQL concepts, i.e. partially correct answers received some marks. The maximum score for each test was 12, with questions being 3, 4, and 5 marks respectively.

After the pretest (first phase), the experimental group went through the pre-action/priming (second) phase prior to the problem solving (third) phase. This was followed by a post-action/reflection (fourth) phase and finally the post-test (fifth) phase. The pre- and post-action phases were both whiteboard sessions with an SQL expert and were each ten minutes in duration. Both these phases *framed* the problem solving phase. In the pre-action phase, the domain expert set the boundary by telling the students the concepts that were to be covered in this session (target concepts). They then progressed through some examples together that were related directly to these concepts. Although the expert directed the pre-action phase, answers, questions, and

comments were elicited from students; this was an interactive phase. The examples, although different to those encountered in the problem-solving phase, were still of varying difficulty, covering the target concepts. The expert also discussed typical misconceptions, and the group was asked to find, work through and discover the reasons for these misconceptions. In the post-action phase, the target concepts were reiterated. Students were prompted to reflect on what they had learnt during the problem-solving phase, including thinking about some of their mistakes and commenting on their own misconceptions. The expert showed the most common mistakes made during that particular problem-solving phase (taken directly from SQL-Tutor) and asked the students to find the mistakes, before collectively working through to reach a correct solution for each.

It could be argued that the pre-tests acted as a part of priming as only questions from the relevant concepts were tested. However, even if this were true, both groups went through the same priming experience with pre-tests.

In contrast, the control group entered the problem-solving (second) phase following the pre-test. They were told that they could improve their database skills by working on problems in SQL-Tutor. The pre-test, problem-solving, and post-test phases were identical for both groups. During the problem-solving phase, both groups worked solely on SQL-Tutor without any assistance from human tutors.

SQL-Tutor has its own strategy for helping the student select the next best problem. It uses a number of factors (such as the student's knowledge of a particular concept, the difficulty levels of the problems solved and to be solved, etc.) to find the *next best problem* for the student. A list of problems within the student's range of capabilities is shown, with the system's choice highlighted; the student has the power to still override the system's choice. We decided to keep this strategy during the problem-solving phase. What is interesting here is that although the Framing strategy was simple in the sense of when it occurred, it still meant that we had partial framework that had multiple strategies working together to help the student achieve the best they could out of the session. Remember here that we are not only testing the validity of the Framing strategy, but the feasibility of having our modules handle multiple and adaptable strategies. Therefore, our curiosity about the

results is more than that of "is Framing a good strategy?".

6.3 Hypotheses

We had two hypotheses for this study:

Hypothesis 1 We believed that since the experimental group were primed for the problem-solving phase, they should be faster in this phase than the control group; i.e. they should have a *higher speed of solving problems* than the control group. The pre-action phase defines the boundaries, provides examples, engages the student in the target concepts, and primes the student in such a way (loading into working memory just what they need) that they are ready for the problem-solving phase. Therefore, the time taken to solve problems should be less.

Hypothesis 2 The experimental group should have higher learning gains than the control group. We postulated that this was due to both the pre-action and post-action phases. Firstly, in the pre-action phase, the student is mentally primed to solve problems. This (according to our first hypothesis) means that they will solve more problems in less time. Secondly, the post-action phase is a reflection phase, where students can think back, solidify what they have learnt and understand and correct their misconceptions. Reflection is known to benefit learning [86, 85, 98]. Both of these situations must have some effect on their learning gains. However, this effect would be minimised as the overall session time was fixed (100 minutes). Therefore students in the control group would have more time solving problems than their counterparts in the experimental group (who would have spent 20 minutes out of the 100 minutes going through the pre- and post-action phases).

6.4 Results

After excluding one student, who did not make any attempts in SQL-Tutor, we had models and logs from 37 students (20 in the control group and 17

Table 6.1: Matched means and standard deviations for test scores (%) and gains.

	Pre-test	Post-test	Gain
Experimental group ($n = 11$)	40.2% ($s.d = 29.1$)	78% ($s.d = 17.6$)	37.9% ($s.d = 31.9$)
Control group ($n = 12$)	47.9% ($s.d = 29.3$)	78.5% ($s.d = 13$)	30.6% ($s.d = 19.6$)

in the experimental). The data we collected and analysed included the pre-/post-test results and just over 43 hours (total) of SQL-Tutor student models and logs in which these students collectively made 2,275 submissions to the system. Remember that a submission is not just one step, but it can include many tasks (or even the full solution).

All students sat the pre-test. There were no significant differences between the pre-test scores for both groups, suggesting that both groups had similar SQL knowledge on the concept targets prior to this evaluation. Not all students completed the post-test. Therefore, Table 6.1 shows the calculated means and standard deviations for both tests and percentage gain only for those students who took both tests (i.e. matched results). Both groups improved significantly ($p < .01$) from pre- to post-test. Although the experimental group increased more than the control group, this was not significant. Further analyses showed that there were no main differences between the groups on individual problems.

The rest of the analyses were done on all 37 students with the data given in Table 6.2. Students in the experimental group on average spent less time solving problems in SQL-Tutor ($60.73\ min, s.d = 14.67$) than those in the control group ($79.34 min, s.d = 33.4$). This is significant at $t(27) = 2.25, p = .016$. This is not an interesting factor as the experimental group spent less time in the problem-solving phase because the overall session time was set to 100 mins. However, what is interesting is that both groups attempted and solved a similar number of problems. The experimental group solved an average of 13 ($s.d = 8$) problems each, while the control group solved on

Table 6.2: Means and standard deviations for experimental and control groups.

	Experimental	Control
Difficulty of problems attempted	4.8 (0.29)	4.93 (0.34)
Difficulty of problems solved	4.79 (0.28)	4.83 (0.36)
Lowest difficulty of problems attempted	3.35 (0.49)	3.5 (0.51)
Highest difficulty of problems attempted	6.4 (1.1)	6.65 (1.22)
Lowest difficulty of problems solved	3.35 (0.49)	3.35 (0.51)
Highest difficulty of problems solved	6.4 (1.0)	6.4 (1.31)
Number of problems solved	13 (8)	12.15 (7.7)
Time spent on problem solving (mins)	60.73 (14.67)	79.34 (33.4)
High-Level Help (HLH) ratio	0.36 (0.22)	0.32 (0.23)
Request for Help (RFH) attempts	2.11 (1.36)	1.95 (1.35)
Relative learning efficiency (E)	0.31	-0.34

average 12.15 ($s.d = 7.8$) problems each. This means that the experimental group solved problems faster than their peers in the control group ($t(26) = 1.8, p = .03$ one tailed, assuming unequal variances). This gave an effect size (Cohen's d) of 0.66 (medium) using the pooled standard deviation.

The number of problems solved was strongly correlated to the number of distinct constraints (domain principles) used by the student, 0.87 for the control group and 0.8 for the experimental group. This means that the control and experimental group saw a similar number of domain facts; these might not necessarily have been the same ones.

The experimental group solved problems faster (the same number of problems in a shorter time) than the control group. But were these problems vastly different? For example, did the experimental group only solve easy problems while the control group solved much more difficult problems? As seen in Table 6.2, the difficulty of problems attempted and solved for both groups was very similar. This even included the highest and lowest difficulty of problems attempted and solved.

How about the usage of help? Did the experimental group use higher amounts of help than the control group and thus move through the problems faster? From the analyses, we can see that the HLH ratio is approximately the same for both groups; 0.36 (0.22) for the experimental and 0.32 (0.23) for

the control group. Both groups requested about the same amount of high-level help. We further looked at Requests for Help (RFH) attempts, as these could have given more help when requested. However, the RFH attempts are similar for both groups. On average, the experimental group made 2.11 (1.36) RFH while the control made 1.95 (1.35).

The relative *learning efficiency* (E) is defined as the performance gained in one condition (e.g. the experimental condition) over the effort expended in relation to another condition (e.g. the control condition). A condition is more efficient if "1) their performance is higher than expected on the basis of their mental effort and/or 2) their invested mental effort is lower than might be expected on the basis of their performance"[139]. To calculate the efficiency of the problem-solving phase of each group, we used "time" as the effort spent and "test gains" as the performance measure. The relative efficiency is found by first converting each of the raw scores to a z score by subtracting the grand mean from the raw score and dividing by the standard deviation. E scores then are found by calculating the perpendicular distance between each z score and the $E = 0$ line when plotted on a Cartesian graph. In this evaluation, the efficiency of the experimental group ($E = 0.31$) was significantly higher than that of the control group ($E = -0.34$); $t(21) = 1.85, p = 0.039$ (one-tailed, assuming unequal variances).

6.5 Discussion

The learning curves for both groups are shown in Fig. 6.3. As can be seen, both graphs are very similar. There are no differences between the learning rates and the fit for curves, which are both high. Looking at the initial knowledge (χ), we can see that both groups had a similar initial domain knowledge. This is confirmed by the similarity in their pre-test scores. Both groups had significant gains between their pre- and post-tests. However, there was no difference between the groups.

In our first hypothesis (see Sec. 6.3) we suggested that students in the experimental group would have a higher speed of learning. This, we attributed, to the pre-action (or priming) phase. Do note that *learning speed* as mentioned here is different to *learning rate* and both should not be confused.

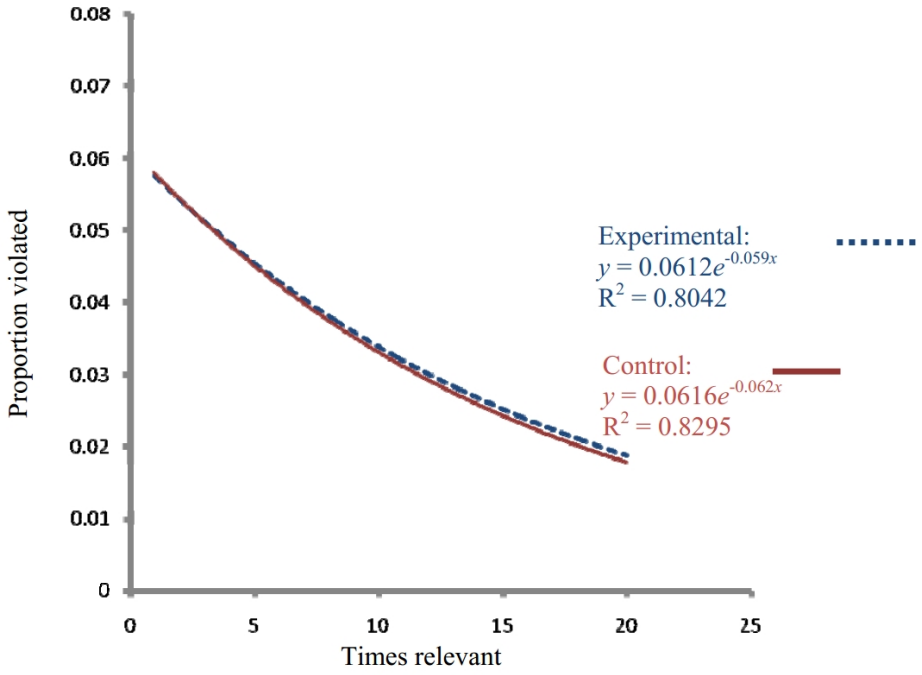

Figure 6.3: Learning curves for both groups

The learning rate is the change in the probability of getting an error for a concept (the rate at which they learnt the concept) per attempt. However, here we are referring to their learning speed, the time taken to learn the concept. Learning speed does not refer to the number of attempts it takes to learn a concept, simply the time.

The pre-action phase helps the student by readying their working memory and loading it with only the essentials required for the session. It also aids by engaging students in active learning about the target concepts. This means that the student is primed and ready for the problem-solving session. The boundaries set on the session reduce the student's problem search space. The pre-action phase helps reduce a student's working memory load, allowing better use of their cognitive resources. These tasks help the student retrieve the needed information quicker while helping the student stay on the current task of solving the problem i.e. it makes the student faster. Note that it does not necessarily reduce the number of attempts it takes for absolute novices to fully solve a problem, but given adequate feeedback, they can link it to the information in their working memory to correct their errors faster. As

these links are created more easily, this could mean that the learning rates for non-novices from the experimental group could increase at a higher rate than their peers in the control group over time.

The experimental group was also significantly more efficient than the control group during the problem-solving phase. In other words, while they did not learn more than the control group, they expended significantly less effort and therefore were more efficient.

A surprising result from this study was that the second hypothesis was not supported. While both groups improved significantly, both groups did so similarly. There are many reasons that one could postulate for this. First, the experiment time may not have been long enough for the experimental group to learn faster to a point where they learnt more than the control group. Second, reflection is usually considered a personal activity; one usually reflects on what they did. Here, we ran the reflection phase as a group activity. This could have affected the effectiveness of the reflection phase. A good question for further research might be to see if group reflections are as effective as individual reflections. Third, not all students might have developed the metacognitive skills required for both the pre- and post-action phases, particularly the ability to connect their errors to the misconceptions while reflecting.

In this Chapter, we were able to take a complex strategy, implement and validate it within the design of our Framework by substituting a human teacher for few of the modules within the Framework. We now believe that this is a valid strategy (at least with the pre-action phase) to keep working with. It also gives us a base to test against when we implement this strategy within our Framework.

Chapter VII

Implementation of the Framework

The results of the study described in this Chapter were published in [105].

In Chapter 6, we used a human teacher to simulate some parts of our Framework. This was both to test the strategy that we were using (to get some sort of a baseline) and also as a proof of concept that our design works.

After that evaluation study, we realised a few things:

- Strategies can be complex. Although many strategies might have a single goal and be simplistic, strategies such as "Framing the Learning Scenario" are quite complex (being made up of groups of strategies, each with their own goal). The Framework needs to be able to cope with not only simple strategies, but groups of strategies working together as a complex strategy. As such, strategies should be usable in other contexts as well. For example, the strategy of self-explanation should be able to be used 1) by itself, 2) within the complex strategy of reflection, and 3) within the even more complex strategy of Framing (in the post-action phase which could include reflection).

- Strategies are made up of Pedagogical Tactics. These are individual, modular, reusable *actions* within the tutor to fulfil a particular small goal. What are the differences between strategies and tactics? Pedagogical strategies always work towards a pedagogically-based goal. For example, keeping the student within a certain HLH ratio is based on some pedagogical belief that this will ultimately affect their learning. However, the goals that pedagogical tactics have are simplistic, small, and not necessarily pedagogically based. An example of a pedagogical tactic is disabling the HLH combo box so that a student cannot request

more HLH. Many pedagogical tactics together work towards the strategy's final goal. Tactics can be reused in other strategies. For example, providing a common misconception is a pedagogical tactic that could be used in both the priming (pre-action) and reflection (post-action) phases of Framing.

- In the study in Chapter 6, the human teacher, being an expert, had a large set of mental resources available to use during the phases. This included worked examples, guided examples, examples with incorrect solutions, problems, tutorial style talks (e.g. talking about the concepts, addressing common misconceptions, etc), etc. Until now, during ITS development, it was up to the ITS developer to add any of these resources if needed into the ITS itself. However, our study showed us that the PSS needs to keep control and know about these resources, so that they can be used as necessary by any strategy. For example, there could be a strategy that states a time when a student should be given a worked example. The PSS needs to know not only that worked examples exist in this resource toolbox, but that an adequate one is available. The meaning (and thus complexity of implementation) of the word "adequate" in the previous sentence depends on the educator.

In the context of the points just made and in relation to this project,

- we have chosen to look at each strategy separately (including within complex strategies) to keep the difficulty level to a manageable level while we create our proof of concept. Even while implementing complex strategies, we will see them as individual strategies, rather than a group. However, with a little more implementation (if this design works), groups of strategies could be implemented making a strategy as complex or simple as necessary.

- we have also implemented a very simple mapping between the pedagogical tactics (actions) and each strategy. This again, is to first get the

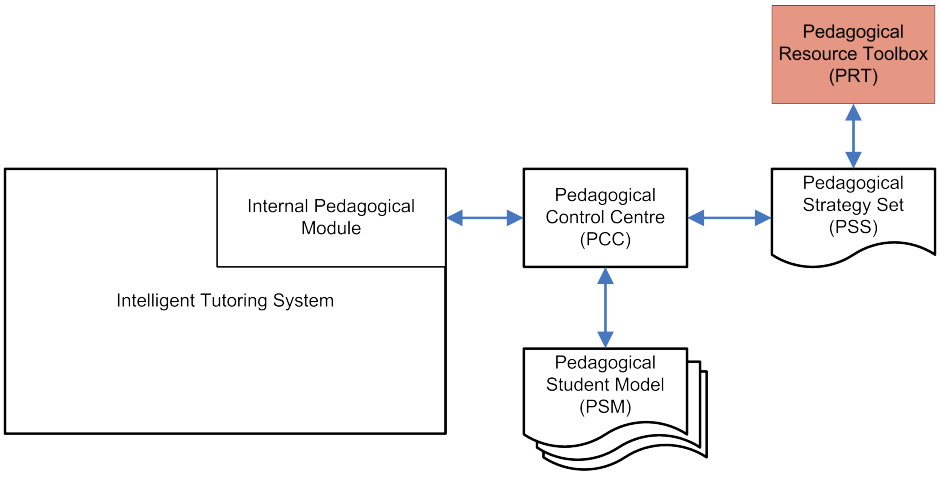

Figure 7.1: Framework design 2

proof of concept working before spending a lot more time on the implementation. In future however, we see a mapping between each strategy and the tactics, laid out as a dependency graph, as some tactics might require ordering while others might not.

- we have separated out the resources from within the ITS and placed them in a Pedagogical Resources Toolbox (PRT). In doing so, we have had to create few new resources. This includes any resource that a strategy might use. This improved design is now shown in Fig. 7.1.

Although this model looked great on paper, it had a few flaws with it, which were discovered during implementation. Keeping it modular was very difficult. The main reason was because many resources in the PRT needed direct access to the Knowledge Base within the ITS. For example, in a CBM tutor, the problems (a resource) have constraints (the knowledge base) that could be relevant to them. Similarly, when giving a worked example (a resource), much of the evaluation of the problem could be done via the evaluator that uses the Knowledge Base. This means that the PSS was now having to be tightly coupled with modules inside the ITS.

Instead of this, we separated the knowledge base from the main ITS. This also has the advantage that it could also be used in a separate situation entirely (separate to the ITS if needed). It means that any modules

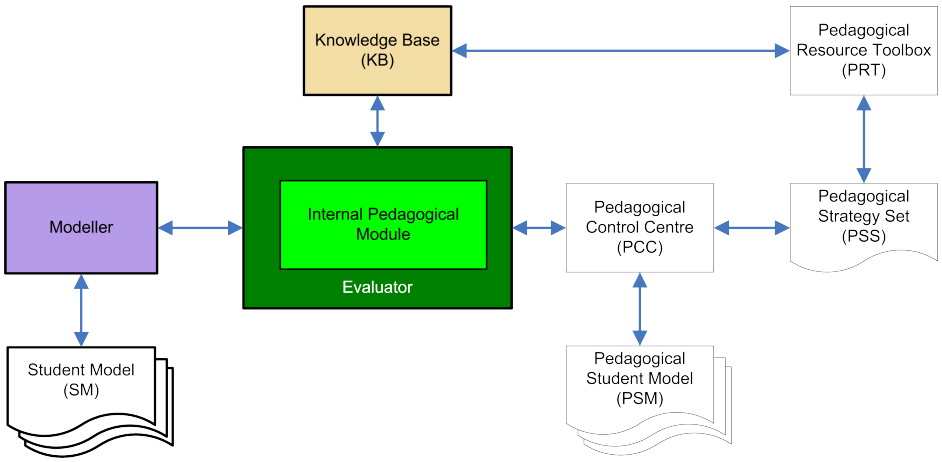

Figure 7.2: Framework design 3

that need it could communicate with it directly. We also separated out the evaluator, within which the internal pedagogical module still resides. This internal pedagogical module is required for domain dependent strategies and feedback. The modeller module's job was a lot simpler now, predominantly dealing with student models. The design (including all the changes) of the framework is given in Fig. 7.2.

Each item in the PRT is directly linked to one or more concepts in the knowledge base. In this project, this link was hard coded when the resource was entered into the PRT; in our case, the link between the resource and the relevant constraints.

Resources vary in granularity. General resources, which cover large portions of the domain are linked to many concepts. These resources are useful for absolute beginners. For example, a tutorial video in SQL-Tutor that covers "the select statement" would probably link to nearly all the concepts (constraints, in this case). This would become less useful for the student as their expertise increased and their need for more specific feedback (via resources) increased. More specific, well-defined resources linking to only a very few concepts would be best, so as to be able to give specific feedback at a particular time when the student is having trouble with that particular concept. The granularity of the resources is left up to the educator.

Pedagogical strategies that require resources are linked to the resource

in the PRT, i.e. the PSS knows that a strategy uses a particular resource and that both must exist for the strategy to work. In this particular case, a tactic would be to show the student the resource. In this project, due to time constraints, we did not implement this logic; it was left up to the developer to know that a resource still existed in the system, thus making sure that all strategies were still valid at all times. In future versions, dependencies would be managed better so that one could not delete a resource without addressing the issues with dependent strategies first.

7.1 Study Design

From our study in Chapter 6, we learnt a number of things:

- The study validated Framing as a potential complex strategy that could be used in the context of an ITS.

- It set a benchmark, albeit one with a human teacher (the ultimate in benchmarks).

- It showed that students in the experimental group improved as significantly as students in the control group, however their speed of learning was faster.

- It seemed as if the pre-action phase was more influential in the positive results than the post-action phase. More research needs to be conducted to understand this aspect better.

- An expert teacher has in their mind a number of resources and we needed to enhance our design to contain a Pedagogical Resource Toolbox which had the ability to contain all these resources to cope with this.

This study acted as a proof of concept to test if our Framework design works. In doing this, we kept things as simple as possible. The ITS we chose was SQL-Tutor. We decided to implement the pre-action phase of Framing and incorporate that with the problem solving phase. As all students were

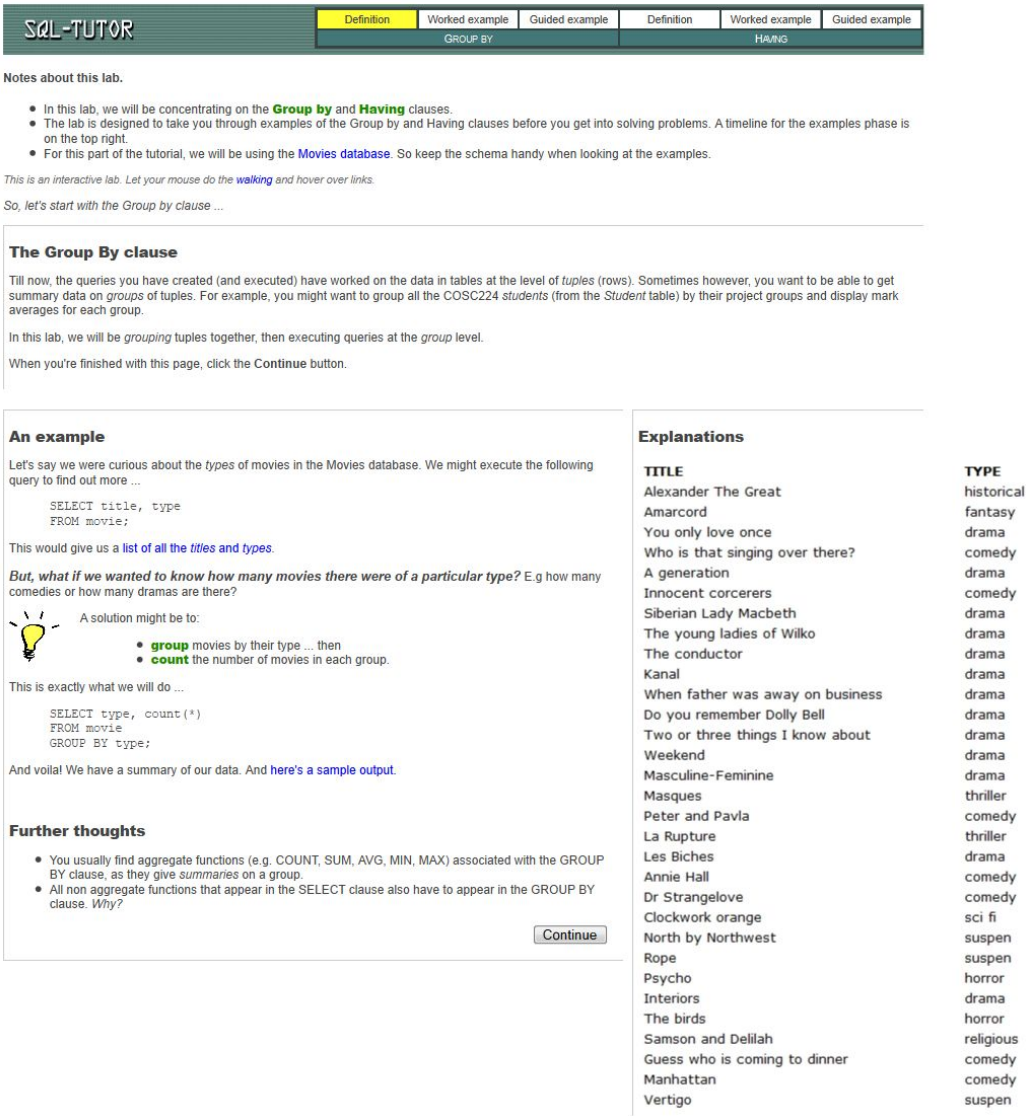

Figure 7.3: The general introduction for the GROUP BY clause. This explanation is given when hovering over the "list of all the titles and types".

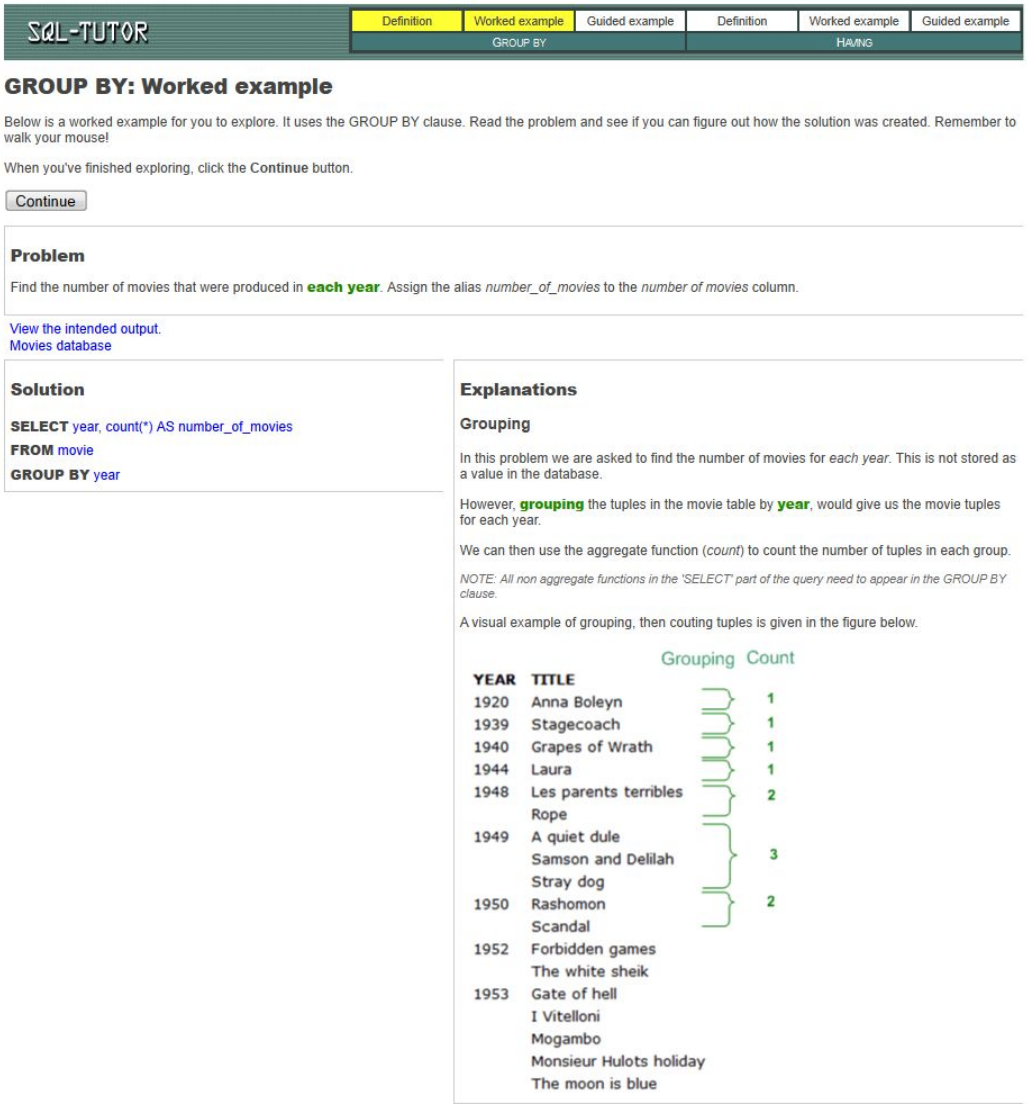

Figure 7.4: The worked example for the GROUP BY clause. This explanation here is given when hovering over "each year".

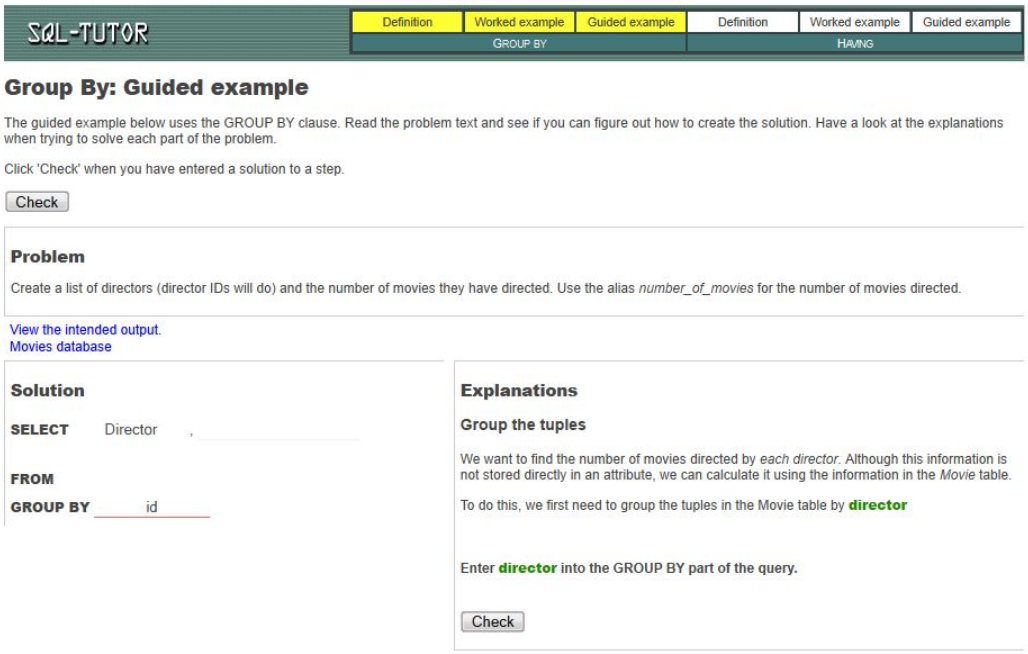

Figure 7.5: The guided example for the GROUP BY clause. This explanation was given after an incorrect attempt at filling in the "Group By" clause.

starting from the same level, we decided to keep the pre-action phase non-adaptive and restrict the target concepts to those required by the SQL clauses *GROUP BY* and *HAVING*. We chose these concepts as the target concepts 1) to keep it similar to the previous evaluation study, and 2) due to the same reasons of choosing them in the first place (e.g. difficult concepts which students overall have a lot of confusion over).

Unlike the human teacher, we did not have a huge variety of resources pre-made for the PRT. Instead we created the following resources, each increasing in active engagement:

1. A general introduction to the concepts (see Fig. 7.3 for the GROUP BY introduction). As mentioned previously, this would be good for any beginners to the topic. Within the explanation, there were hyperlinks that when the student hovered over, provided additional explanations or examples. The introduction page also gave an example with explanations and a possible solution from a real database in SQL-Tutor. A *Further thoughts* section at the end gave more information to extend

their knowledge of the clause. In Fig. 7.3, the student is up to the step in the example where the example is only half done (it is missing the GROUP BY clause). They have hovered over the hyperlink to see what their output would be at this stage. The output given here is for the first query that is partially done. This would then give them a better idea of exactly what the GROUP BY clause does, particularly when they hover over the final link to see the output to the full solution.

2. A worked example (see Fig. 7.4 for the GROUP BY worked example). This is where an example (containing the problem and its solution) from the target concepts was presented. Each part of the solution was implemented as a separate hyperlink. Students could hover over any of the hyperlinks to receive detailed explanations for that part of the solution. Hovering over each part of the solution also highlighted the relevant part of the problem statement, allowing students to link the problem to the solution. For example, when hovering over the GROUP BY part of the Select statement (as seen in Fig. 7.4), participants not only saw the explanation of the grouping concept, but also an example from a real database in SQL-Tutor on how the grouping affected the solution. Students could also click on links to view the intended output of the query, or view the database schema.

3. A strictly guided example (see Fig. 7.5 for the GROUP BY guided example). This is different to the worked example in that it is more active than the worked example. Here, a student was given a problem with a "fill in the blanks" style empty solution. When a student clicked on any of the blanks, they got hints and explanations for that part of the solution. However, they were required to complete the solution themselves correctly before being able to proceed any further. If a student was stuck on part of the worked solution, they simply had to click 'check' again without making any further changes, and it bottomed out and gave them the answer for that particular step with explanations; in Fig 7.5, the student has bottomed out on the GROUP BY clause and has been given the answer for that step (following the explanation).

The student could click the "Check" button to check their solution.

This pre-action phase was conducted for both the GROUP BY and the HAVING target concepts. Figures 7.6, 7.7 and 7.8 show the interface for the pre-action phase of the HAVING clause. The strategy here was simple: if a student had not attempted any questions from the target concepts, then provide them with the pre-action phase first, before they solve problems on their own. Sub-strategies ensured that each section of the pre-action phase was completed correctly before they could move on.

7.2 Hypothesis

In our earlier study, we used a human teacher to simulate significant aspects of the Framework. In education, human teachers are held as the gold standard against which we test our systems. We found that both groups improved significantly. There was a difference in speed of learning, where the experimental group was both faster and more efficient at solving problems. Through this, we realised that Framing (particularly the pre-action phase) is a viable strategy to implement.

In our current version of the system, we do not have a human teacher intervening at any stage of the process. We also have a limited set of resources, compared to the human teacher. These resources were not made adaptive either (although they all happened during the pre-action phase). Furthermore, we chose not to deal with misconceptions as this could cause confusion if not done well. With all these limitations, our hypothesis is that students in the experimental group should do at least as well as the students in the control group. If done correctly, we should at least see trends in the correct direction.

7.3 Results

Thirty students from COSC226 (the second-year database course at the University of Canterbury) participated in the evaluation for no monetary reward. They were divided randomly into two groups: control and experimental. The students participated in the study during their usual lab ses-

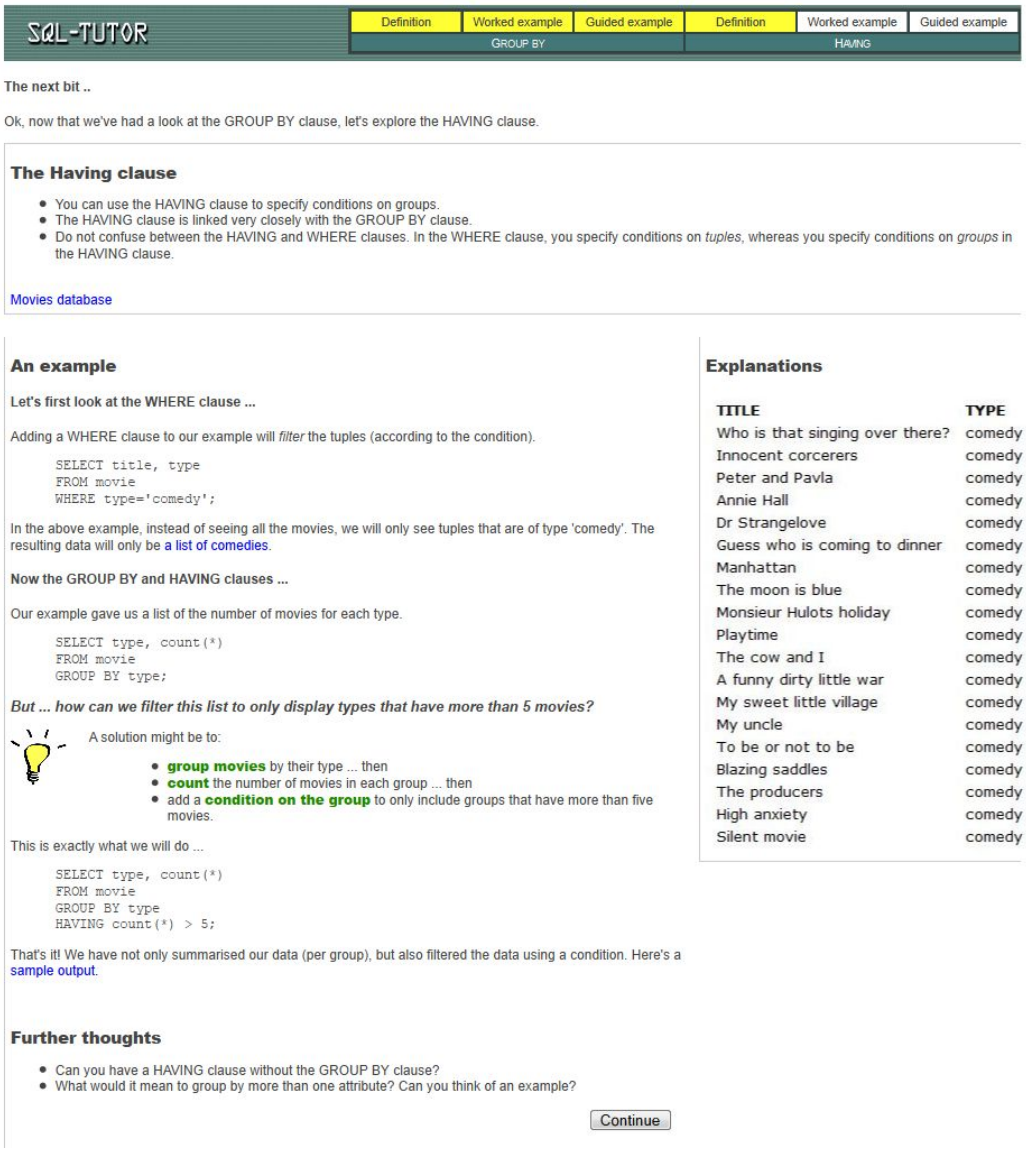

Figure 7.6: The general introduction for the HAVING clause. The explanation given here is when hovering over "list of comedies".

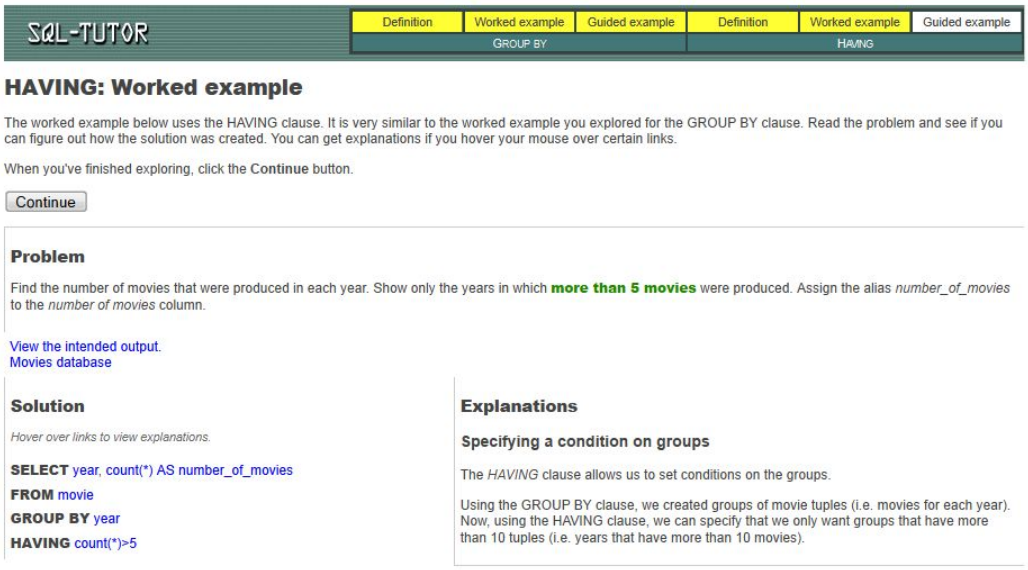

Figure 7.7: The worked example for the HAVING clause. This explanation is given when hovering over the "Having" clause.

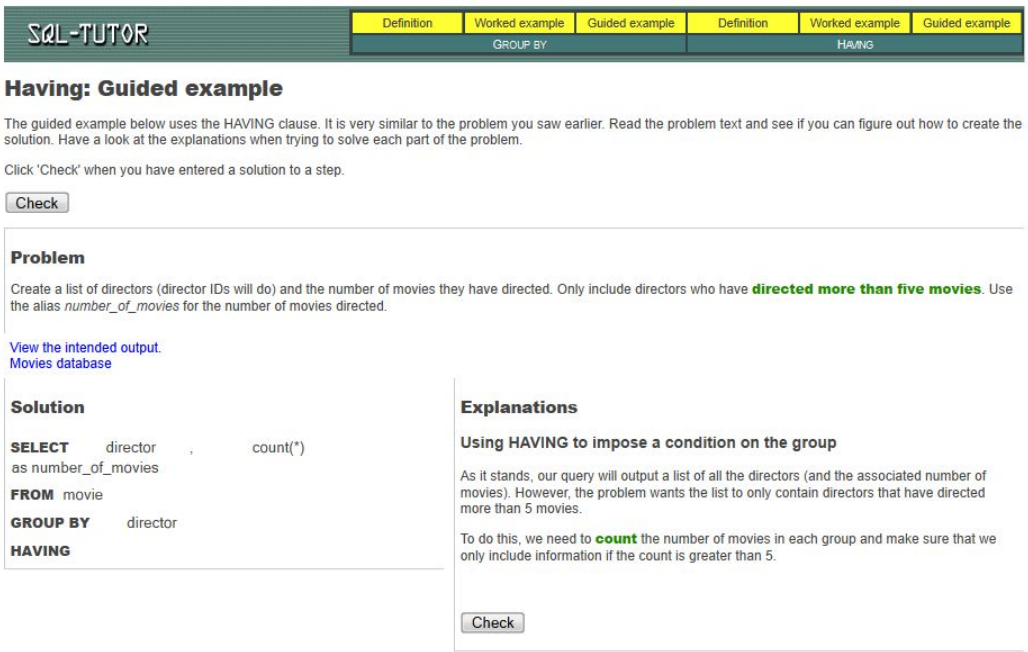

Figure 7.8: The guided example for the HAVING clause. This explanation is given after the student's current solution was checked.

Table 7.1: Matched means and standard deviations for test scores (%) and gains.

	Pre-test	Post-test	Gain
Experimental group ($n = 5$)	57.14% ($s.d = 39.7$)	90.44% ($s.d = 16.6$)	33.3% ($s.d = 39.5$)
Control group ($n = 12$)	52.9% ($s.d = 26.3$)	88.3% ($s.d = 11.2$)	36.6% ($s.d = 24.7$)

sions for this course. The lab sessions are generally held sometime after the lectures (containing the material pertinent to the labs). Two lab sessions were used for this study. Each lab session controlled for time to 100 minutes (total) and were held on 13 and 14 May 2009. All students completed the pre- and post-tests at the beginning and end of their sessions respectively. These tests were exactly the same ones used in the previous evaluation study (see Chapter 6). The experimental group had four phases (pre-test, pre-action, problem-solving, post-test) while the control group had three (pre-test, problem-solving, post-test); all phases were sequential.

The data we collected and analysed included the pre-/post-test results and just over 29 hours (total) of SQL-Tutor student models and logs in which 30 students collectively made 1,769 submissions to the system. There were 17 students in the control group and 13 in the experimental. However, only 17 students sat both tests, and we give the matched results in Table 7.1.

There were no significant differences between the two groups on pre-tests, post-tests, and gains. All students improved significantly between the pre- and the post-test, showing that they improved in their domain knowledge during the session.

The rest of the analyses were carried out on all the thirty students and the results are reported in Table 7.2.

As mentioned earlier, these results provide us with trends. As expected, the trends in this stage, although not as pronounced, were very similar to those found in the earlier study (with the human teacher). The experimental group spent less time solving problems; this was marginally significant;

Table 7.2: Means and standard deviations for experimental and control groups.

	Experimental ($n = 17$)	Control ($n = 13$)
Difficulty of problems attempted	5.02 (0.59)	4.95 (0.53)
Difficulty of problems solved	5.01 (0.55)	4.91 (0.56)
Lowest difficulty of problems attempted	3.53 (0.51)	3.64 (0.49)
Highest difficulty of problems attempted	7.0 (1.35)	6.82 (1.7)
Lowest difficulty of problems solved	3.61 (0.50)	3.364 (0.49)
Highest difficulty of problems solved	6.92 (1.32)	6.7 (1.82)
Number of problems solved	10.15 (5.03)	10.5 (6.4)
Time spent on problem solving (mins)	52.46 (18.09)	65.17 (33.9)
High-Level Help (HLH) ratio	0.46 (0.26)	0.43 (0.34)
Request for Help (RFH) attempts	1.84 (0.68)	1.88 (1.40)
Relative learning efficiency (E)	0.11	-0.12

$t(25) = 1.3, p = .09$. Participants in both groups solved a similar number of problems. This means that the participants in the experimental group solved problems faster than those in the control group; this was also marginally significant; $t(19) = 0.46, p = .09$).

We performed very similar analyses to those in the earlier evaluation study.

Did the students in the experimental group attempt or solve easier problems? If so, this might account for the differing speed of problem solving. The problems attempted and solved were of similar difficulty level on average. This was also confirmed for the highest and lowest difficulty level of problems attempted and solved in both groups i.e. participants solved similar types of problems.

What about the number of attempts and feedback that the students received? On average, the experimental group made 49 (26.6) attempts while solving problems, while the control group made 69 (49.3) attempts. The difference was not significant showing that both groups got similar amounts of feedback from the system. However, to check that one group did not receive higher levels of help from the system, we checked the High-level help (HLH) ratio [103, 107]. As defined earlier, the greater the HLH, the more feedback

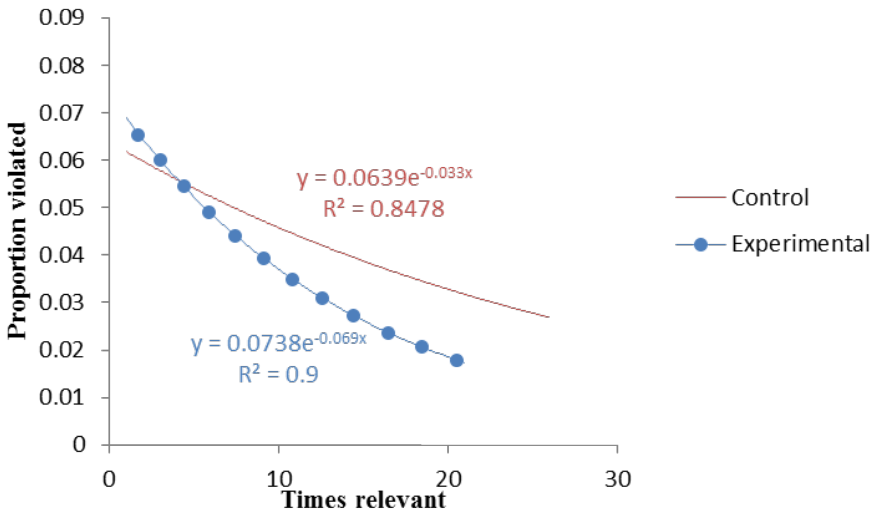

Figure 7.9: Learning curves for the experimental and control groups

a student receives that would help them directly solve the problem; the full solution, for example, is the highest HLH. Another important characteristic in SQL-Tutor is that the HLH has to be requested manually by the user (i.e. it does not automatically increment). From the analyses, participants from both groups used similar amounts of HLH during this phase; 0.46 (0.26) for the experimental group and 0.43 (0.34) for the control group. The Requests for Help (RFH) are another way of eliciting more help from the system. Here, there was no significant difference between the experimental (1.84 (0.68)) and the control (1.88 (1.40)) groups.

Next we checked the relative learning efficiency (E), just as we did with the previous evaluation study. Again, we used "time" as the effort spent and "test gains" as the performance measure. As with the previous study, the efficiency of the experimental group ($E = 0.11$) was higher than that of the control group ($E = -0.12$). This was marginally significant ($t(28) = 1.11, p = 0.1$, one-tailed, assuming unequal variances).

We also plotted learning curves for both groups (see Fig. 7.9) using data from all thirty students. Although the differences were not significant, the trend lines indicate that the experimental group learned at a higher rate than

the control group.

This Chapter shows us that we received the same trends as we did for the study in the previous study, in spite of the limitations. We also improved on our basic design for the Framework.

Chapter VIII

The Evolution of the Framework

In Chapter 7, we created a new design for the MAPS Framework. We also took apart the ITS itself and divided it into various interconnected modules (see Fig. 7.2) to work with MAPS. Even though it worked well in our study (and we believe it would in similar instances), there is yet another group of situations that we have not yet accounted for; this is when the graphical user interface itself has to change when a particular strategy is applied.

In the current design, all the strategies would have to work within the confines of the current user interface. This Chapter deals with the evolution of the Framework, so that it can deal with other user interfaces dependent on the strategy that is being applied. Here, we are not referring to the changes of the user interface that occur normally as a part of the activity, but with graphical user changes that occur purely because another strategy is applied. As an example, NORMIT has several steps to solve a particular problem. Each step is on a different "page" and each page's user interface looks quite different to the other steps. However, this is its normal mode of operation, regardless of the strategy chosen. These are not the graphical user interface changes we refer to in this chapter. However, if we were to apply the worked examples strategy, we would need the user interface to swap from the problem-solving interface to the worked examples interface; this would be purely due to the change in pedagogical strategy. In our current framework, from the previous chapter, we have not allowed for this. This chapter deals with such scenarios.

8.1 Erroneous Examples as a Strategy

A relatively under-studied strategy is erroneous examples. This is when the teacher provides the student with a worked example that has errors in it; the student has to detect the error and fix the incorrect solution. There have been studies done when correct solutions are presented to the student, both with and without ITSs (see Section 3.2.5). Most work with erroneous examples in computer based teaching environments has been conducted within the ActiveMath [110] learning environment.

There are many benefits for using erroneous examples [109]:

- This particular task could foster self-explanation.
- Detecting an error is a valuable skill in itself.
- It stimulates reflection and exploration.
- Finding the error could be a source for inquiry-based learning.
- Working on failures can help the student practice critical thinking.
- The focus of the learner shifts from performance-oriented learning towards learning-oriented learning, which in turn leads to deeper knowledge. Problem-solving could push the learner towards performance-oriented learning.
- Finding the error, then looking for alternative methods of solving it, pushes the student to use their meta-cognitive skills.
- Debugging a solution is a similar strategy that novice human tutors have to go through. That skill (of debugging errors) is known to produce learning.

Promising results have been found from using this strategy in studies done in ActiveMath [172, 171].

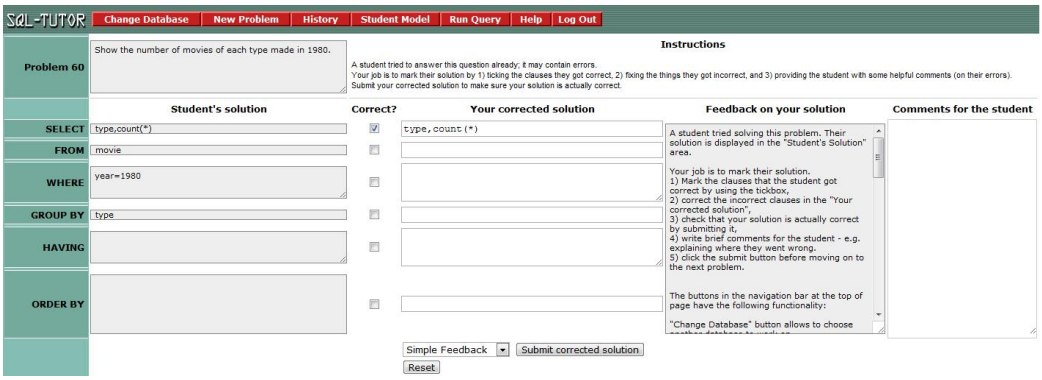

Figure 8.1: SQL-Tutor interface for erroneous solutions

8.2 Evaluation study

As with the previous study, this study acted as a proof of concept to test if the MAPS Framework works with these new requirements. We are not concerned about whether this strategy itself works, but whether we could make enough modifications in the MAPS Framework to allow changes to the user interface dependent on strategy. Again, we kept things as simple as possible. We worked with SQL-Tutor as the base ITS as well.

The participants in the study were students from a second year database course at the University of Canterbury. They attended lectures on SQL prior to their lab, during which they used SQL-Tutor. Participants were randomly selected into experimental and control groups. They were told that the outcomes from this study did not count towards their grades; no reward or payment was given for participating in the study. Students used the system they were assigned to for at least 3 weeks.

The new version of SQL-Tutor assigned students a particular strategy: either the problem solving strategy (the control group), or the erroneous worked examples strategy (the experimental group). The graphical interfaces are quite different to each other. Dependent on the strategy, the student saw the relevant interface. We created it, such that if at any time the strategy was changed, the interface would follow suit. Once students were assigned a strategy, they sat a pre-test. Students who were using the problem solving strategy saw the interface shown in Fig. 2.3, while students who were in the

Table 8.1: Means and standard deviations for control and experimental groups.

	Control (n=35)	Experimental (n=31)
Pre-test	1.9 (0.9)	2.03 (0.75)
Attempted problems	30.97 (37.6)	16.78 (19.2)
Solved problems	28.9 (37.6)	15.51 (19.2)
Total time spent (mins)	172.02 (207.4)	77.39 (77.9)
Min difficulty of problems solved	1.03 (0.18)	1.24 (0.51)
Max difficulty of problems solved	6.83 (2.41)	6.2 (2.24)
HLH Ratio	0.50 (0.21)	0.36 (0.27)
Lab test result (%)	59.6 (18.3)	58.6 (17.6)

erroneous worked examples strategy saw the interface in Fig. 8.1.

In the problem-solving interface, the problem text is displayed at the top of the screen with an area below it for the solution workspace where a student could enter their answer. The erroneous examples interface is slightly more complex. The student is told that they are marking a student's solution (not their own). The problem text is still displayed on the top of the screen. However, this time, there are two solution workspaces displayed; the first contains the erroneous solution and is disabled to editing. The student checks each of the clauses and decides whether the clause is correct. If it is correct, they tick the "correct" checkbox which automatically enters that clause into the second (editable) solution workspace. If they decide that it is incorrect, they type the correct clause into the second workspace. There is also an optional area where they could give textual feedback to the "student". We concluded that at the very worst (if the student ignores the erroneous solution completely), the student would be still solving problems. At the very best, the student would be detecting and correcting the errors in the erroneous solution; furthermore, they would be explaining the error to the student via the optional text box. In both groups, when the student

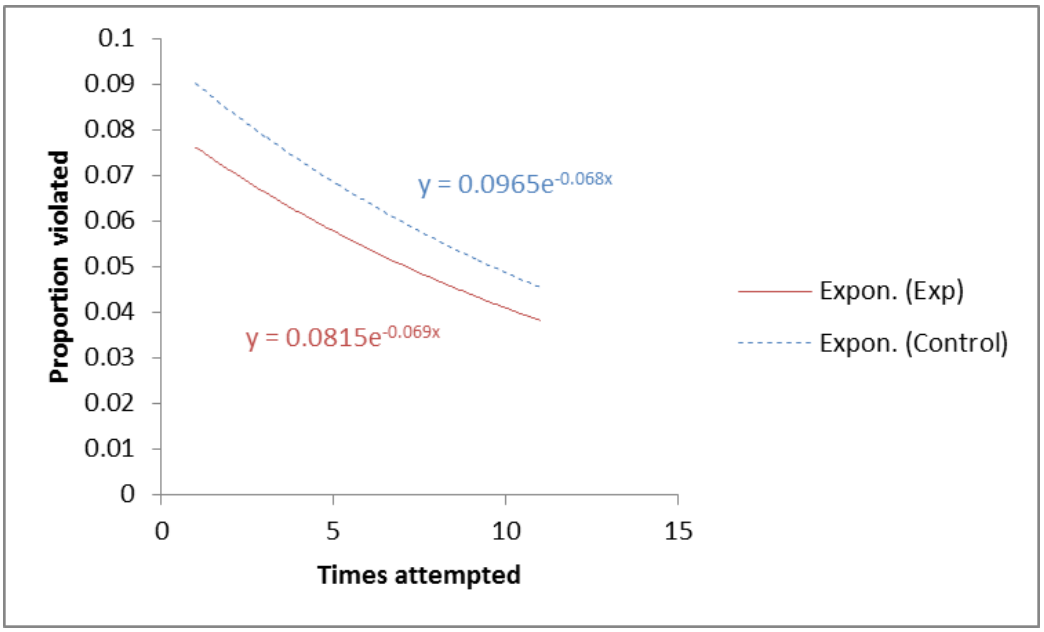

Figure 8.2: Learning curves for the study on erroneous solutions

was ready, they could submit the solution to the system to get the system feedback on their solution.

There was no significant difference between the control (1.9 (0.9)) and the experimental (2.03 (0.75)) groups on the pre-test, meaning they came from the same population; the marks were out of 5. Unfortunately, not many students took the post-test, and so those results are not included here. The rest of the data analyses include all students and are given in Table 8.1. Towards the end of the evaluation period, students sat a lab test in which they were asked to form SQL queries. The lab test marks are included in here just for mild comparison, but should be viewed with caution. Firstly, the queries they learnt in SQL-Tutor were a subset of what was given in the lab test. Secondly, students could have possibly used other means (other than SQL-Tutor) to study for the lab test.

From the analyses in Table 8.1, we can see that the experimental group took significantly less time ($t(44) = 2.5, p < 0.01$), and attempted and solved approximately half as many problems as the control group. This is understandable, as correcting others' mistakes takes up significantly higher cognitive load and "marking" each problem is time-consuming. Both groups tried

and solved similar types of problems (with similar difficulty levels). The experimental group requested significantly less amounts of High Level Help than the control group ($t(56) = 2.2, p = 0.016$). Looking at the learning curves in Fig. 8.2 (which included data from all students), we can see that both groups learnt well, with the experimental curve significantly different to the control ($t(25) = 2.63, p < .01$). In spite of the experimental group spending much less time in SQL-Tutor, both groups did equally well on the lab test. Again, although this is a good result, we caution the reader about putting too much emphasis on the lab test results.

8.3 Changes to the Architecture of the MAPS Framework

For the interface to be able to change dependent on pedagogical strategy, we need the strategy itself to know, and then inform the system, what kind of interface it needs. This means that it has to have a link with the communications module. The communications module takes care of any communication between the user and the system; this includes user interfaces. Sometimes, the strategy might call for other parts of the current interface to be changed, such as having an embedded movie file. This movie file, along with other resources, is kept in the Pedagogical Resource Toolkit (PRT). One could argue that the interfaces should also be in the PRT. However, we have defined the PRT as having resources only (i.e. things that, in themselves, provide information, or help the student in some way). The interface itself does not do this; it is a communication mechanism. Hence, we placed it in the communications module with the other interfaces. The final architecture of MAPS is given in Fig. 8.3. The main difference that we made is that the communications module now has a number of interfaces. Each strategy has the opportunity to choose what interface it uses.

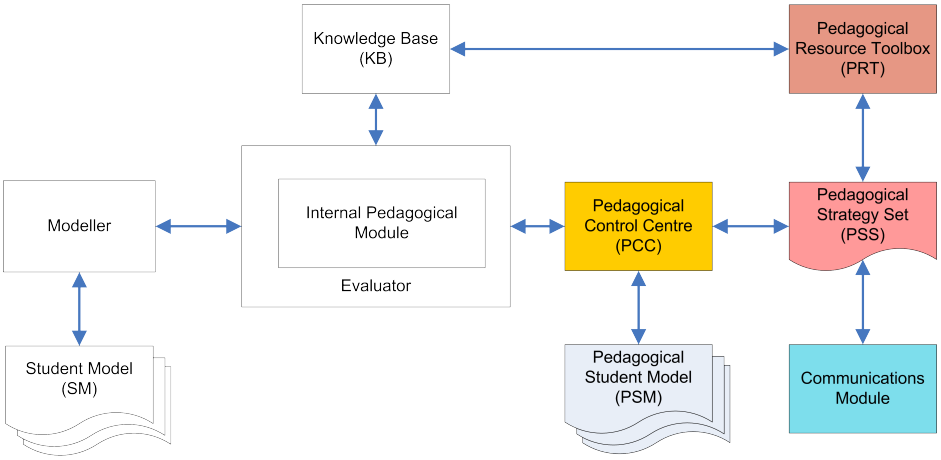

Figure 8.3: Architecture of the evolved MAPS Framework

Chapter IX

Beyond the Framework Design

This Chapter is akin to an enhanced version of a *Future Work* section, where we can seriously ponder the future of this Framework design. We base our vision on current work, make some assumptions along the way, and allow ourselves some leeway in visualising what could be.

Several researchers have envisioned a tutor that encompasses a much larger domain space, often referred to as the *mega tutor* [131, 151, 117, 39]. The mega tutor would cover a greater domain than a single ITS. One way this might be achieved is by having several tutoring systems interconnected in such a way that they all share at least a portion of the domain and student knowledge between them. The student logs into the mega tutor once and seamlessly progresses through the various tutors, while the tutors share a combined knowledge about the student.

So far, we have concentrated on designing a framework whereby educators can implement their own strategies within a tutor. In this Chapter, we provide two examples of work done in combining knowledge between tutors and extrapolate our Framework design using results from these examples. The first example is a component that we created called the SQL-Tutor Resource Component (STRC). The second example is the combination of student models between two separate ITSs, namely EER-Tutor and ERM-Tutor, using an application manager we created called *Overwatch*.

9.1 STRC: SQL-Tutor Resource Component

As mentioned above, in certain versions of the mega tutor, multiple tutors work together to provide a seamless experience for the student by sharing information from certain modules among themselves; *Adapt*^{2} [41] is one such

example. In *Adapt*^{2}, the various adaptive web-based systems that are connected together combine and share their student models using a global modelling server. The global modelling server understands a global ontology. Each system that uses this modelling server has to map their own ontological language to the global ontology. This is a manual process and can be time consuming [189]. Using this method, as the student moves between tutors (without even realising they are doing this), *Adapt*^{2} is able to use combined knowledge about the student and provide more adaptive assistance with each interaction.

In this section we explore one of the tutoring systems (STRC) that we created that is connected in such a way to *Adapt*^{2} [159].

STRC is a modified version of SQL-Tutor that enables it to act as a component within a mega tutor or in conjunction with another external server (e.g. another tutoring system). The architecture of STRC is shown in Fig. 9.1.

The STRC is not a stand-alone application like SQL-Tutor. It relies on some external server requesting services from it. In the STRC, only internal pedagogical strategies that work within a problem are kept. This means that the STRC gives feedback on student solution submissions but does not for example, choose the next best problem for the student to work on. The overall high-level pedagogical strategies are left to the external server. While the STRC maintains the short-term student model, the external server keeps the combined long-term model of the student over all the interactions (including those from other tutoring systems).

SQL-Tutor is a constraint-based ITS, and students are modelled based on their knowledge of constraints. To have a common language between the external server and STRC, a global ontology of the domain (SQL) was created by experts [160]. A mapping was then created between the tutor's language (i.e. constraints) and the global ontology [161]. The STRC has a module that uses this mapping to map from constraints to the common ontology; it is a many-to-many mapping. All communications between the external server and STRC are done using the mapped models. However, each server keeps their own versions of the student models.

A communication protocol was established so that the STRC could com-

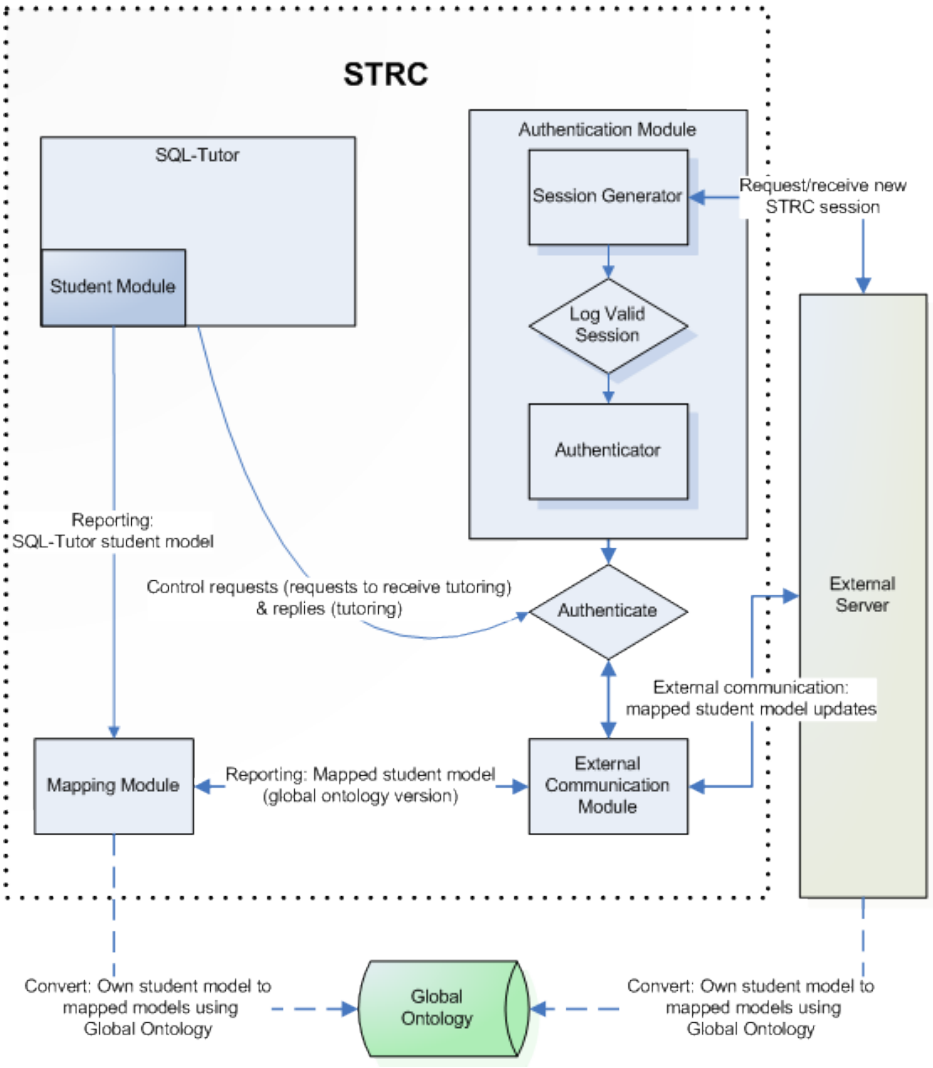

Figure 9.1: STRC Architecture, adapted from [159]

municate with any authenticated external server. On all communications, the external server first authenticates with the authentication module in the STRC. After successful authentication, a valid session is created and the student could work on problems from the STRC. This is all done behind the scenes, so is seamless from the point of view of the student.

After each student solution submission, the STRC maps the current short-term student model relating to the student's solution in terms of the global ontology and sends this information to the external server. The external server updates its modelling server when it receives information from each tutor.

In the current version of this system, the student controls their path through the activities, choosing what topics and questions on which they would like to focus. *Adapt*^{2} modifies links and target knowledge icons to let the student have more information about their knowledge, allowing them to make a more informed decision. This mega tutor has been trialled successfully in real classrooms [159, 40].

We believe that our Framework design would work well with this sort of a system. See Fig. 9.2 for the architecture of this type of mega tutor but with the components of the Framework. For simplicity in the picture, we have excluded the authentication and external communication modules (see Fig. 9.1), which would still be part of the tutoring system component.

In this version, the student models in the tutoring system component goes through a process of mapping using the global ontology before it is communicated to the main central server and stored in the global modeller. This is the same as what STRC and *Adapt*^{2} do currently. However, the pedagogical elements of the framework are overlayed between the main central system and the tutoring system. In this case, the PCC advises the tutoring system on what strategies to use. It chooses these strategies from the PSS, based on the student's pedagogical student model and the logic within the PCC. A pedagogical student model is stored for each student on the central server and forms a part of the global modeller. This gives a more complete picture of the student, not just at the domain level, but also at the pedagogical level. The resources in the PRT are tagged according to concepts in the global ontology. The tutoring system knows what resources to ask for, as there is also

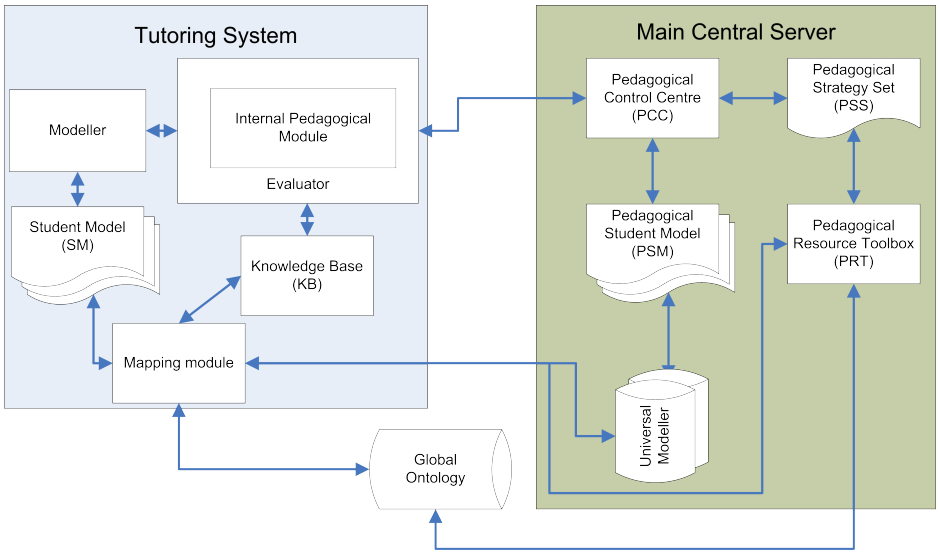

Figure 9.2: Architecture of a central server connected to one tutoring server

a mapping between the local knowledge base and the global ontology.

This version of the Framework would allow, not just domain knowledge, but pedagogical strategies as well to be shared amongst the multiple tutoring systems connected to the main central server.

9.2 Overwatch: Sharing the student model between two ITSs

EER-Tutor [120, 165, 190] and ERM-Tutor [112] are both constraint-based ITSs created and run by the Intelligent Computer Tutoring Group at the University of Canterbury. Both tutors teach subdomains within the overall domain of databases. EER-Tutor gives students the opportunity to practice Enhanced Entity Relationships while ERM-Tutor provides students with practising their Entity to Relational Mapping skills. EER-Tutor works with an ill-defined task where students can start at any place within the solution process while ERM tutor works with a very well-defined set of sub-tasks. Both these tutors are and have been used in university database courses for a number of years. They have also been through several evaluation studies. When students are learning about databases, they generally learn to create Entity Relationship diagrams (using EER-Tutor) and then move onto learning how to model the diagrams for the particular database (using ERM-

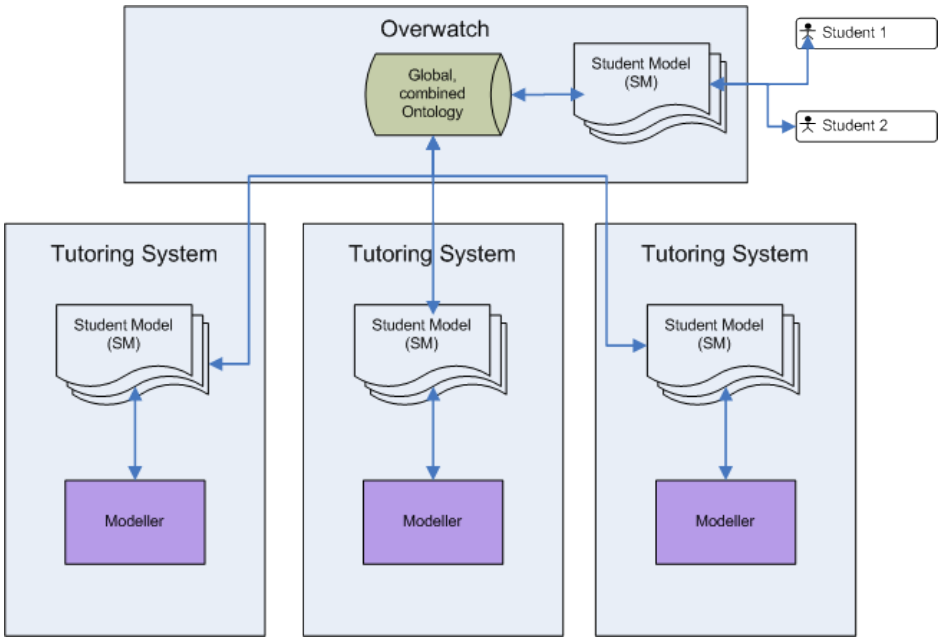

Figure 9.3: The Overwatch application combines models from different tutors

Tutor).

As these tutors are separate and stand-alone applications, the first interaction with a student on either tutor is truly the first interaction between both parties. Even though there is an amount of overlap between the two domains, the tutors do not pass any information about the student from one tutor to another. This is a waste as each tutor treats the same student as a new one, starting their student models from scratch. If the first tutor could pass on information about the student's learning to the next tutor, the second tutor would be better prepared and the knowledge about the student would simply grow as more evidence was added to their student model.

During the summer of 2011/12, we had a project within the ICTG to create a new learning environment, encompassing both tutors, but where the student's knowledge within the domain is shared between both tutoring systems [135]. This also affected the visualisations of the student model, so that when the student started using the second system, their visualisations changed accordingly, rather than starting from the beginning.

This was done by creating a centralised and independent application manager called *Overwatch* (see Fig. 9.3). Overwatch contains a global ontology,

which is the combination of the ontologies of all tutors. When a new tutor is plugged into Overwatch, their ontology has to be added to the combined global ontology manually. Overwatch also contains the global student modeller. This modeller bases its student model on the global ontology. Overwatch is event driven; continually keeping an eye on the student models of each of the ITSs plugged into it, updating the student's global student model appropriately as changes are noticed.

Using this method, the student can move between tutoring systems seamlessly and their global student model is used throughout, including when visualising open student models.

In this particular case, both tutors were constraint-based tutoring systems. As such they were not as disparate as they could have been. However, the manual process of combining ontologies was still time consuming as the constraints between the tutors were quite different.

We could overlay our Framework on this design so that the tutoring systems are not only sharing domain knowledge, but pedagogical knowledge as well. As with the Framework design, each tutor still has a smaller internal pedagogical module to deal with the details of that particular domain and tutor, but the high level domain independent strategies could be shared via a common PSS and PCC. The PSS, PCC, PRT, and PSM could be housed within Overwatch. Student models could also be held within Overwatch, not just about their domain knowledge, but about their pedagogical knowledge making it a much more complete model. As students move between tutors, the new tutor would already know what types of pedagogies worked with this student previously, what did not work, and what has not yet been tried.

Chapter X

Conclusions

Until now, ITSs have predominantly adapted their tutoring based on a student's domain knowledge (i.e. their student model). Using this process, ITSs have boasted large learning gains even during short learning sessions. However, we still have a long way to go before reaching those learning gains achieved from the one-to-one student-to-teacher scenario. To this goal, researchers have been looking at strategies used by good human tutors to try different pedagogical strategies within ITSs and measure their effects on learning. Most of this has been done in controlled studies where one group (the experimental or treatment group) use a version of the ITS with the new pedagogical strategy and the other group (the control group) uses some previously benchmarked version of the ITS or condition. Several such studies are continuing to be conducted (e.g. [149],[152], [122]). These studies are very valuable as they tell us something about the pedagogical strategy in question and inform us about previously unknown educational research and human behaviour.

Studies such as the ones mentioned above are beneficial for research. However, preparing a tutoring system for such a study usually requires quite a lot of work by developers as they change the underlying structure of the ITS to meet the requirements of the new strategy. More importantly, as ITSs become more common and are used in the classrooms, the ITS must have the ability to not only adapt based on the student's knowledge of the domain, but also on pedagogical strategy. This is a primary difference between the ITSs of today and a good human tutor. A good human tutor has a set of pedagogical strategies they can use for the various learning points, which they can adapt dependent on many factors, e.g. the current expertise of the student, whether a strategy worked before, whether a strategy follows

a particular learning theory, etc. A typical ITS today does not have this ability, and for good reason; it is very difficult to implement such an ITS.

ITSs have mainly based their architecture on some version of the architecture given in [30], which includes the domain knowledge model, the student model, the communications module, the pedagogical module, and sometimes the expert model. These versions have worked well to adapt to the domain knowledge of the student. However, we believe that a different architecture needs to be applied so that multiple pedagogical strategies can be implemented for each learning point, and furthermore they can adapt depending on relevant factors.

This has been the main goal of this project: to see if we could develop a framework that would allow for multiple, adaptable pedagogical strategies. In doing so, we divided the pedagogical section of the ITS into a module that holds the strategies (PSS), one that holds all the resources used by the strategies (PRT), and one that holds the logic to control the selection and adaptation of the strategies (PCC). The PCC gets its information from both the normal student model and the pedagogical student model, which holds important information about each student, but in terms of pedagogical strategies. We also discussed and designed a way that a pedagogical strategy could use particular user interfaces, ones that are different from the normal user interfaces used by the ITS; this was in its interconnection with the communications module. We called this new design the MAPS (Multiple, Adaptable, Pedagogical Strategy) Framework.

The method used to design this Framework was incremental, with studies in real classrooms conducted along the way, comparing the new system (with a new strategy) to an already well-known version of the ITS.

How do we intend this Framework to be used?

1. **Educators** can specify multiple pedagogical strategies using pedagogical tactics (the building blocks of a strategy). For example, they could say that they want students to be kept within the 0.4-0.6 HLH Ratio; they could specify how a new topic should be introduced; or they could specify when worked examples should be shown to the students. At the moment, our strategies are written in the form of constraints, however,

we do not say that this is the only way of coding the strategies. For example, they could be written as rules, specifying what action must be taken when a particular situation is encountered and a particular goal is intended. Being entirely separate from the domain, they could be written in any appropriate language.

2. **Researchers** can use this to specify more than one pedagogical strategy for a point of learning. This way, they can study the interactions between multiple strategies, rather than just comparing one strategy to another (which is also easily possible using this Framework). This also raises the benchmark for comparison purposes.

3. **Students** could use this to alter the pedagogical strategy by which they are taught. We have not developed the interface for this, however, with this Framework, it is possible. How much should students be involved in their learning? Is giving them the ability to change the pedagogical strategies by which they are taught a good thing?

One of the noticeable things about this Framework is that it breaks the whole system into modular parts. While until now each tutoring system has been an entity in its own right, the MAPS Framework differentiates at the component level. This means that, if done correctly, these parts could even be shared between tutoring systems. For example, pedagogical strategies themselves (e.g. the PSS) could be shared between tutoring systems. This might not always be the correct thing to do (i.e. different domains might require different strategies), however, with this Framework it is possible. One could imagine a case where the components were spread across the semantic web. One researcher could easily then use the resources from another PRT. In other cases over the semantic web, researchers could incorporate each others' pedagogical strategies to work in their tutoring systems. Knowledge bases could be shared via global ontologies. This could then lead to many exciting possibilities, such as truly distributed tutoring systems that merge into each other (i.e. the student can go from one tutoring system to another seamlessly). We have seen the early stages of this in Chapter 9.

10.1 Limitations

There are a number of limitations of this research; most, simply because they lie outside the scope of this project.

The PSS does not come with a set of pre-defined pedagogical strategies. We have left this to the educator using the tutoring system to decide. On one hand this design gives them full freedom, whereas on the other, they do not have a quick starting point to begin using the tutoring system. We believe that it is up to the educator and not the computer scientist to specify the correct pedagogical strategies or to try new ones.

The logic within the PCC is beyond the scope of this project. In this project, a pedagogical strategy was chosen from a matching set using a very simple algorithm (the top-most relevant strategy). This area opens up a large research field into how strategies should be chosen. In fact, instead of choosing just one strategy at a time, there might be times where it is suitable to choose multiple strategies at once.

The division between low-level domain-dependent and high-level domain-independent strategies is again left to the educator. The one bit of advice we offer is that any pedagogical strategy that directly deals with specific domain principles should go into the internal pedagogical module; those that can be generalised in any way should go into the PSS.

As mentioned earlier, we tried the MAPS Framework only in the context of SQL-Tutor, a constraint-based ITS. In spite of this, we believe that the Framework is independent enough to work well, regardless of the domain modelling method. However, we have not tried this with a non constraint-based ITS.

Given more time, we would have liked to have created a user interface for creating and changing strategies. Unfortunately this still has to be done in code.

There are still some dependencies between some parts of the system, i.e. not all components are independent of each other. For example, each resource in the PRT is linked in some way to the knowledge base. In future, we would like to reduce the dependencies by creating Application Programming Interfaces (APIs) between each component.

Given these limitations, we still believe that the MAPS Framework can provide many benefits, not simply of pedagogical strategies, but also of collaboration and sharing between tutoring systems.

Chapter XI

Publications

This section contains the publications related directly to this project.

Mathews, M., and Mitrovic, A. Do students who see more concepts in an ITS learn more? In *1st International Conference on Educational Data Mining* (Montreal, Quebec, Canada, 2008), R. S. J. d. Baker, T. Barnes, and J. E. Beck, Eds., Educational Data Mining, pp. 266-273.

Mathews, M., and Mitrovic, A. How does students' help-seeking behaviour affect learning? In *Intelligent Tutoring Systems*, B. Woolf, E. Aimeur, R. Nkambou, and S. Lajoie, Eds., vol. 5091 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2008, pp. 363-372.

Mathews, M., Mitrovic, A., and Thomson, D. Analysing high-level help-seeking behaviour in ITSs. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, W. Nejdl, J. Kay, P. Pu, and E. Herder, Eds., vol. 5149 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2008, pp. 312-315.

Mathews, M., and Mitrovic, A. Does framing a problem-solving scenario influence learning? In *Proceedings of the 17th International Conference on Computers in Education (2009)*, S. C. Kong, H. Ogata, H. C. Arnseth, C. K. K. Chan, T. Hirashima, F. Klett, J. H. M. Lee, C. C. Liu, C. K. Looi, M. Milrad, A. Mitrovic, K. Nakabayashi, S. L. Wong, and S. J. H. Yang, Eds., pp. 27-34.

Mathews, M., and Mitrovic, A. Incorporating framing into SQL-Tutor. In *Workshop Proceedings of the 19th International Conference on Computers in Education (2011)*, A. Mohd Ayub, B. Chang, and K. Leelawong, Eds., pp. 391-398

Do Students Who See More Concepts in an ITS Learn More?

Moffat Mathews and Tanja Mitrović
{moffat, tanja}@cosc.canterbury.ac.nz
Intelligent Computer Tutoring Group, Computer Science & Software Engineering,
University of Canterbury

Abstract. Active engagement in the subject material has been strongly linked to deeper learning. In traditional teaching environments, even though the student might be presented with new concepts, it is possible for the student to remain passive to such an extent that it is detrimental to learning. This research explores whether experiencing new concepts in an ITS necessarily equates to learning. Initial analysis of data mining student models in SQL-Tutor, a CBM tutor, shows a strong positive correlation between the number of constraints seen and the number of constraints learned. This global trend is mitigated at an individual level, possibly due to individual differences in learning style and behavior. The number of constraints not learned remains relatively constant for all students; however, the proportion of constraints not learned is inversely proportional to the constraints seen. The author suggests deeper analysis into the factors that might cause variability amongst individuals from this population trend.

1 Introduction

For many years, researchers have been stating the importance of active participation and engagement in learning [1, 2]. Constructivists would argue that for effective learning to occur, the student has to be actively involved in constructing the appropriate mental model of the domain, ideally guided by the teacher [3]. Learning which requires the student to actively construct their knowledge leads to better performance (both in time and the number of important errors) [4]. Students who work at constructing their own knowledge using methods such as self-explanation, become better problem solvers [5]. In contrast, students that passively sit in a traditional class environment learn poorly compared to those that are actively involved in constructing their knowledge [6].

It is easy to understand the scenario of a student who learns very little in a traditional lecture environment even if many new concepts are introduced, because they are disengaged from their learning (e.g. daydreaming). However, does this same principle apply in ITSs? Can students use an ITS and learn nothing because they are so passive with their learning? Basic observation would suggest that there are differences between the way a student learns in a traditional lecture environment and within an ITS. In the traditional lecture environment, the student can be passive (in spite of being presented with new concepts) to the extent that they learn nothing. For example, an extreme case might mean they could have slept through the lecture. The progress through material, including the rate at which it is delivered, is usually dependent on an external source (e.g. the lecturer) rather than the student. However, in an ITS, the student generally must initiate all progress i.e. to be presented with anything, the student has to do something. This might be as small as selecting a new problem or requesting help. Disengaging totally

from the learning would mean not progressing through the ITS. There usually is no external entity built into the ITS that controls and maintains the steady flow and delivery of knowledge. Progress is student-dependent. Due to this basic difference, one could assume that if a student progresses through problems in an ITS (in any manner), they are at least in some way actively involved in some part of their learning. As active participation is correlated to learning, students in an ITS must learn more as they progress through the system.

The aim of this research is to mine student logs in an ITS to see if students learn more as they see and experience more concepts or domain principles in the ITS. The goal of this paper is to complete the first stage of this research i.e. to use simple statistics to find general trends within populations. Once these trends are established, in-depth analysis can be performed.

In the next section we introduce the ITS used, namely SQL-Tutor. After describing the dataset, we outline the methods used to perform this analysis. We finish with a discussion of the results and future work.

2 SQL-Tutor

The data used for this research was gathered from SQL-Tutor [7], an Intelligent Tutoring System. It provides students with intelligent and adaptive guidance as they practice their skills within a specifically designed problem solving environment for the domain of database querying using Structured Query Language (SQL). It has been used by students in tertiary database courses since 1998. Having undergone several evaluation studies, it has been shown to produce high levels (both rates and depth) of learning [8].

The domain in SQL-Tutor is modeled using constraints. Constraints are domain principles. They contain a relevance condition and a satisfaction condition. The relevance condition is the condition in which the constraint is applicable (or is relevant). The satisfaction condition states what must be true for this constraint to be correct. For example, for the domain "driving", one constraint might state "if you are driving in New Zealand, you must be on the left-hand side of the road." The relevance condition informs us on when this constraint applies: *"if you are driving in New Zealand"*. If this constraint is relevant, then for it to be correct, the satisfaction condition must be true, i.e. *"you must be on the left-hand side of the road"*. See [9] for a detailed explanation of Constraint-Based Modeling.

Each time a student attempts a problem in SQL-Tutor, the constraints that are relevant for the attempt are recorded in the student model. For each relevant constraint, a history of correct usage for each attempt is also recorded.

SQL-Tutor contains approximately 280 problems. Each problem is assigned a difficulty level by the teacher, using their expertise in the domain. Difficulty levels range from 1 (easiest) to 9 (hardest). For each solution to a problem, several constraints could be relevant i.e. each problem could relate to several concepts. The number and complexity of the relevant constraints also determines the difficulty of the problem. For example, a

problem with many relevant constraints, or a problem with complex relevant constraints could be said to contain many principles or complex principles respectively, making the problem more difficult than one with fewer, simpler constraints. This means that when solving more difficult problems, the student could be dealing with either a large number of constraints or more complex constraints.

3 Data Used

The data for this analysis was collated from student models and logs from an online version of SQL-Tutor. Students from all over the world were given free access to this version of SQL-Tutor when they bought certain database textbooks. Students that made less than five attempts were excluded from this analysis. The final dataset consisted of 1,803 students who spent just over 1,959 combined hours while making a total of 104,062 submissions. In total, these students solved just over 70% (19,604) of the problems they attempted (27,676).

Certain constraints (such as basic syntactic constraints) are always relevant for every query i.e. those concepts apply to every problem. As these constraints are principles that are very easily learned and do not provide in-depth knowledge into the domain or its concepts, they were excluded from this study. The number of constraints that students saw at least once while solving problems varied from 6 to 333. While engaged in their problem solving activities, students experienced a combined total of 174,309 constraints.

4 Method

The method utilized was very simple. We first extracted constraint histories from individual logs and student models. Using this data, we counted the number of unique constraints that were relevant for each student. These are the constraints that the student saw or experienced during their practice; we listed these as "constraints seen".

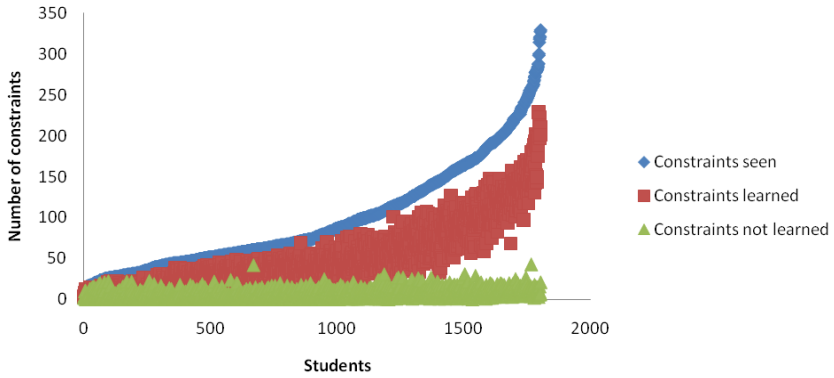

Figure 1: Constraints seen, constraints learned, and constraints not learned for each student, ordered by constraints seen.

To calculate whether a constraint was learned, we used two separate methods. In both methods, we used a window to focus on the recent usage of each constraint. We used the most recent history as we determined that this would better indicate the current state of

learning with regards to that particular constraint. The main difference between the two methods was the size of the window.

In the first method, if the total history of a particular constraint consisted of five or more attempts, we used a window size of five. If the history consisted of less than five attempts, we used a window size of three. In the second method, we always used a window size of five. Each method makes an assumption of how many attempts are required to determine the state of learning for each constraint.

We then calculated the proportion the constraint was learned by dividing the number of correct usages of the constraint in the window by the window size.

$$Proportion\ constraint\ learned = \frac{Correct\ usages\ of\ constraint\ in\ window}{Window\ size}$$

This gave us a number between zero and one, where zero meant that the constraint was not learned at all whereas one meant that the constraint was learned. All other numbers in between zero and one showed the proportion the constraint was learned. We plotted graphs for all users depicting their number of constraints seen, the number of constraints learned, and the number of constraints not learned, using both the methods described above. Both methods above gave very similar graphs. Figure 1 shows the values from method 2. We used these methods as they were previously used somewhat successfully in various versions and evaluation studies.

We also calculated the average difficulty level of problems attempted and problems solved for each student. This is shown in Figure 2.

The total time spent on the system by each student was recorded and graphed against the number of constraints seen (Figure 3). Furthermore, we calculated the *time per constraint seen* and the *time per constraint learned* for each student (Figure 4). Time per constraint seen was calculated by dividing the total time by number of constraints seen. Time per constraint learned was similarly calculated by dividing total time by the number of constraints learned. These calculations were to give us an idea of how much time each student spent in relation to the constraints they had seen (how quickly they were progressing through the system) and to the constraints they had learned (how quickly they were learning constraints).

5 Results and Discussion

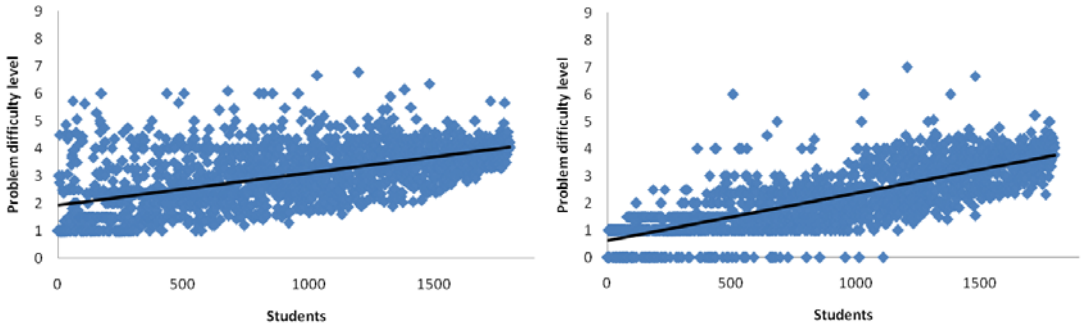

Figure 2: Average difficulty levels of problems attempted (left) and solved (right) for all students ordered by number of constraints seen.

5.1 Constraints

As seen in Figure 1 and from correlation calculations, the number of constraints a student sees is strongly correlated to the constraints learned ($Pearson's\ r = 0.947$). This means that the students who saw more domain principles learned more. This supports our original hypothesis that learning on an ITS requires some form of active engagement which translates to learning. Remember, here we are saying that to have *seen* a constraint, they must have made at least an attempt where that constraint was relevant. Simply moving through and viewing the problem is not counted as progressing through the system as the student would not have experienced any constraints i.e. they have not made any attempts. The variability in the constraints learned could be attributed to individual differences. These differences could be due to the quality and quantity of engagement, the amount of help used, gaming [10], and low skill levels (e.g. low meta-cognitive abilities).

The number of constraints not learned remains relatively constant for all users, regardless of the number of constraints seen. This is interesting, but makes sense as the proportion of constraints that are wrong decreases as the number of constraints seen increases. However, it seems that the types of constraints that are not learned are different between users. Even though all students are getting approximately the same number of constraints wrong, the students who have seen more constraints are dealing with more difficult constraints. This can be seen from the graphs in Figure 2. Students who had low numbers of constraints seen (to the left of each graph) are generally attempting and solving easier problems compared to those who have seen higher numbers of constraints (to the right of each graph). As mentioned earlier, this means that the students on the right of the graph are using more complex constraints or a greater number of constraints in a single problem than the ones on the left.

5.2 Time

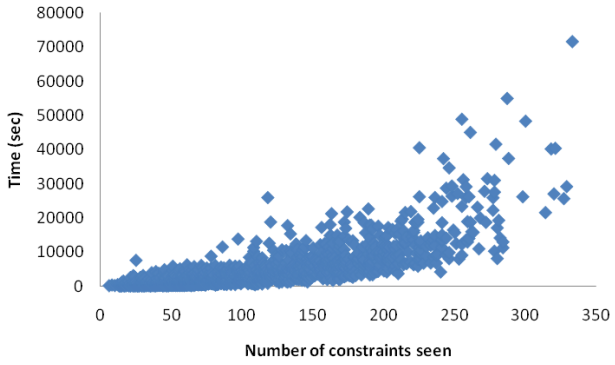

Figure 3: Number of constraints seen against total time taken by students

Figure 3 shows the number of constraints seen plotted against the total time spent in the system. From the graph, we can see that the total time is proportional to the number of constraints seen i.e. the students that saw more constraints spent longer in the system. This is understandable as the greater the number of concepts explored by the student, the longer they will spend in the system. Although this may seem a trivial point, it is included for completeness.

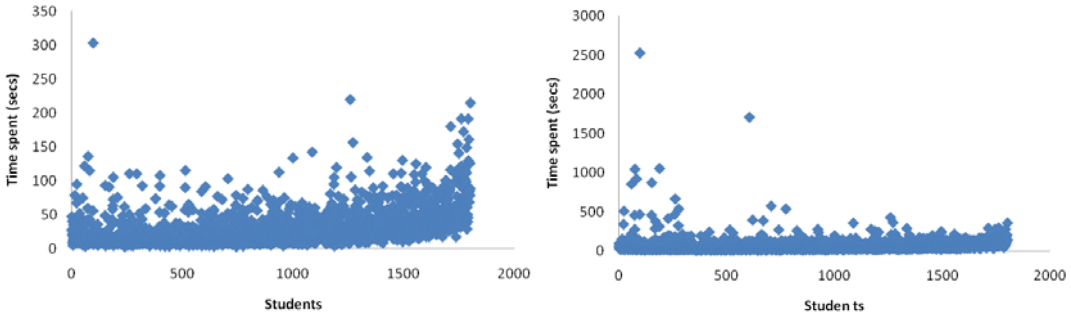

Figure 4: Time spent per constraint seen (left) and constraint learned (right) for each student

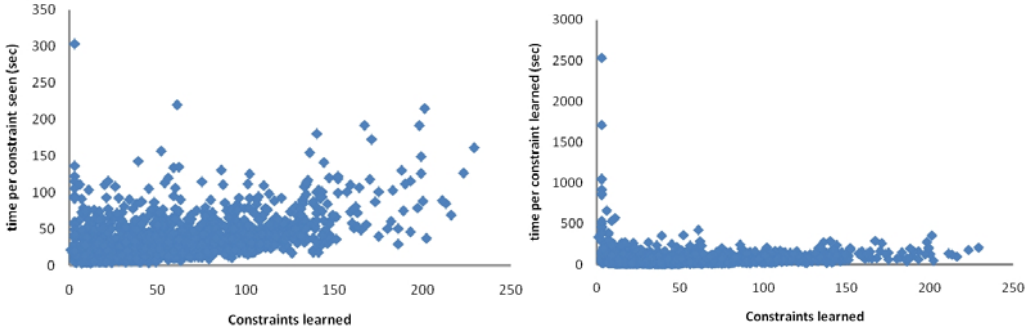

Figure 5: Time spent per constraints seen (left) and constraints learned (right) against constraints learned

The time spent per constraint seen and the time spent per constraint learned (Figure 4) is relatively constant across students, irrespective of the number of constraints seen or learned. In Figure 4, the students who have seen the least number of constraints are towards the left of the graphs. This means that even for the relatively difficult constraints, students on average still spent the same amount of time on each constraint. This could be because students that are seeing more constraints are generally the ones that have progressed to higher levels of expertise. The outliers could be students who are attempting more difficult problems than their expertise level. Similarly, Figure 5 shows that the time spent per constraint seen is relatively constant (or increases slightly with high variability) as more constraints are learned. The time per constraint learned is relatively constant as the number of constraints learned increases. This means that although students spent varying amounts of time on the ITS (depending on the number of constraints they saw), they spent on average the same amount of time per constraint.

6 Conclusion and Future Work

This paper looked at the broad global trends within a population of students using an Intelligent Tutoring System (SQL-Tutor). On a global (population) level, there is a strong positive correlation between the number of constraints a student sees and the number of constraints they learn within an ITS. This seems to support our initial hypothesis that those students who progress through problems in an ITS do learn concepts as they require at least a certain amount of active participation. However, at a more localized level, there are students who have seen many constraints but have learned far fewer constraints than their counterparts who have seen fewer constraints. This could be due to individual differences in the way they learn (e.g. their styles of learning) or in their behavior (e.g. gaming the system). More detailed analysis is required to understand what factors play a part in this process, including factors that affect causality.

We would like to perform deeper analysis on the types of constraints that the students do not learn. What concepts in SQL do they not understand? Are there similarities between these constraints? Are there certain constructs that are inherently more difficult to such a point that no students understand them? Are students guessing their answers and therefore seeing constraints that are not necessarily relevant for the problems they are solving? The answers to these questions would give us better insight into the domain and the way in which students learn concepts.

Another part of this research which requires further exploration is the method by which we calculate constraints learned. In our research, we used two separate methods which gave very similar results. However, *"what does it mean to have learned a constraint or principle?"* is still a very valid question. How much of the constraint history should we use to determine if the constraint has been learned?

In the introduction, we made a comparison between the student in the traditional classroom environment and a student working on an ITS. We said that the extremely passive student in a traditional classroom might not learn anything while the student working on an ITS has a higher chance of learning concepts as they see more concepts. However, it should be noted that the extremely passive student might be one that lacks

motivation to use the ITS. In fact, we did not include anyone who made less than five attempts in our dataset. In this research we do not make any mention about motivating students to learn; merely the trends found in students that do use ITSs.

In spite of these points, this initial study is encouraging, indicating the effectiveness of learning for students that progress through the material in ITSs.

References

[1] R. S. Prawat, "Teachers' Beliefs about Teaching and Learning: A Constructivist Perspective," *American Journal of Education*, vol. 100, pp. 354-395, 1992.

[2] F. N. Akhras and J. A. Self, "System Intelligence in Constructivist Learning," *International Journal of Artificial Intelligence in Education*, vol. 11, pp. 344-376, 2000.

[3] M. Ben-Ari, "Constructivism in Computer Science Education," presented at SIGSCE, Atlanta, GA, USA, 1998.

[4] R. Catrambone and M. Yuasa, "Acquisition of procedures: The effects of example elaborations and active learning exercises," *Learning and Instruction*, vol. 16, pp. 139-153, 2006.

[5] M. T. H. Chi and K. A. VanLehn, "The Content of Physics Self-Explanations," *The Journal of the Learning Sciences*, vol. 1, pp. 69-105, 1991.

[6] L. B. Resnick, "Constructing Knowledge in School," in *Development and Learning: Conflict or Congruence?*, L. S. Liben, Ed. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 19-50.

[7] A. Mitrović, "An Intelligent SQL Tutor on the Web," *International Journal of Artificial Intelligence in Education*, vol. 13, pp. 173-197, 2003.

[8] A. Mitrović, B. Martin, and P. Suraweera, "Intelligent Tutors for All: The Constraint-Based Approach," *IEEE Intelligent Educational Systems*, vol. 22, pp. 38-45, 2007.

[9] S. Ohlsson, "Constraint-Based Student Modelling," in *Student Modelling: The Key to Individualized Knowledge-based Instruction*, vol. 125, J. E. Greer and G. I. McCalla, Eds. Berlin: Springer-Verlag GmbH, 1994, pp. 167-189.

[10] R. S. Baker, A. T. Corbett, K. R. Koedinger, and I. Roll, "Detecting When Students Game the System, Across Tutor Subjects and Classroom Cohorts," presented at UM 2005, Berlin, Heidelberg, 2005.

How Does Students' Help-Seeking Behaviour Affect Learning?

Moffat Mathews and Tanja Mitrović

Computer Science & Software Engineering,
University of Canterbury, New Zealand.
(moffat,tanja)@cosc.canterbury.ac.nz

Abstract. We examined high-level help (HLH) seeking behaviour of students by data mining in SQL-Tutor. Students who used HLH very frequently had the lowest learning rate; their learning was also shallow. They attempted very difficult problems compared to other groups but only solved very easy problems, suggesting that they were usually situated well beyond their Zone of Proximal Development. They also abandoned a large number of problems without solving them. Manual inspection of the logs showed erratic problem solving behaviour, suggesting a "guess and copy" strategy.
The group of students who used HLH very infrequently seemed to contain two distinct sub-groups: students with high expertise, and students with very low expertise who still did not use HLH. Learning rates were highest for students who used moderate HLH. Students with lower usage of HLH solved the most difficult problems comparatively, without the use of HLH, and had high learning rates, suggesting the ITS is most beneficial for this group of students.

Key words: Help-seeking behaviour, data mining

1 Introduction

This paper presents the initial results of data mining student models and log files in SQL-Tutor [1]. The research conducted in this paper is the first in a series of steps towards a project designed to develop a general framework for adapting pedagogical strategies in Intelligent Tutoring Systems (ITSs). Part of the challenge of designing such a framework is to identify the various contexts in which students find themselves when working in an ITS, and understand their behaviour when confronted with such a context. As researchers and developers, we are also interested in the effects of this behaviour on learning. Research is continuing into various aspects of help seeking behaviour [2], including investigating the misuse of help and feedback in ITSs [3, 4]. In this step of the research, we explore one aspect of help-seeking behaviour in students: the frequency of high-level help (HLH) sought and its effect on learning.

Help in an ITS usually consists of tutor-specific help and domain-specific help. Tutor-specific help is help regarding the ITS itself. For example, this type

of help might show the student steps on how to submit a particular problem, or how to request a new problem. In contrast, domain-specific help concentrates on concepts within the domain irrespective of any tutoring system used.

Domain help can be categorised in many ways. One way is to class it into either problem-dependent or concept-dependent help. Examples of problem-dependent help are hints or feedback messages provided for a particular problem or step within that problem. These can either be given before the problem is attempted (forward hints) or after a solution has been submitted (feedback). Concept dependent help is help on domain concepts and is usually given in the form of tutorials or feedback during a meta-cognitive dialogue (e.g. during reflection or self explanation).

Domain help can also be categorised into adaptive or non-adaptive help. Adaptive help is customised for the particular student, and is based on the student's solution or their student model. An example of non-adaptive help is the full solution.

For this research, we have also categorised domain help into low-level help (LLH) and high-level help (HLH). Low and high level refer to the degree of help given. For example, showing the student a partial or full solution is classed as high-level help, whereas telling the student that they have made some errors is classed as low-level help. Sometimes, the process of giving higher degrees of help till the highest degree is reached is referred to as "bottoming out". The highest degree of problem-dependent help is usually the full solution, either for the step or the entire problem. In this paper, we focus on the use of LLH and HLH to explore its effect on learning.

2 Problem-Dependent Help in SQL-Tutor

SQL-Tutor is an ITS designed to guide and support students in their deliberate practice in the domain of querying databases in Structured Query Language (SQL). SQL-Tutor has a long history of high learning rates and evaluation studies, and is used in tertiary database courses as part of the curricula. Being a constraint-based modelling tutor, it stores domain knowledge as constraints (domain principles) and provides the student with multiple levels of domain-help depending on various factors, such as their current solution, their student model, and the constraints violated on this submission.

In SQL-Tutor, the problem-dependent, domain help is divided into six numbered categories. These categories in order of degree of help are: 1. Simple feedback, 2. Error flag, 3. Hint, 4. Partial solution, 5. List all errors, and 6. Full solution. The help level (also known as feedback level) is controlled on the interface by means of a simple combo box (see Fig. 1). At the start of a new problem, the tutor defaults to help level 1. On subsequent incorrect submissions, the tutor automatically increments the help level by one, to a maximum of 3 (i.e. hint). The student can override the help selection at any time by changing the value of the combo box. Help levels 4, 5, and 6 have to be specifically requested by the student. For this research, we have classed help levels 1-3 (inclusive) as

LLH, and help levels 4-6 (inclusive) as HLH. Furthermore, LLH is automatically incremented, but HLH is requested by the student.

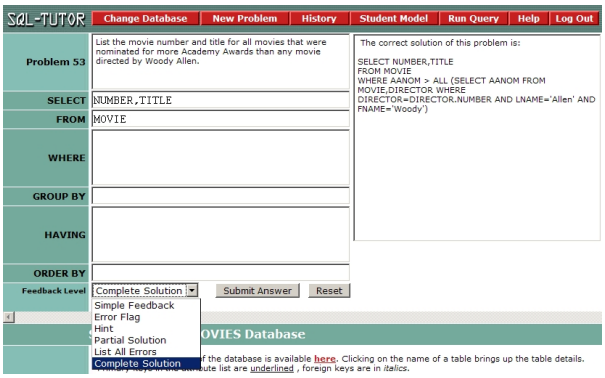

Fig. 1. The SQL-Tutor task environment showing the combo box with the various levels of problem-dependent help. Here, the student has requested HLH (in this case, the full solution) for the problem, which is displayed in the feedback pane.

When a student reaches the task workspace (Fig. 1) in SQL-Tutor, they are presented with the problem in text format. More information regarding the context of the problem, such as information about the schema, is also presented with each problem. The solution workspace contains text areas in which students can work on their query. Once the student is content with their solution, they can submit it and receive feedback. The degree to which this feedback is given is dependent on the help level selected, as discussed above. It is this help - and more specifically, the difference between LLH and HLH used by students, that interests us for this paper.

3 Learning Curves

Learning curves are used to plot students' learning over time. Learning a skill generally follows a power law, where the greatest improvement occurs early in the learning process. The formula for the power law is given in equation 1. More details about learning curves can be found in [5, 6].

$$E = \chi n^{\alpha}. \tag{1}$$

Where: E is the error rate, χ is the performance on the first trial, n is the opportunity to practice the skill, and α is the learning rate.

Four main points are worth noting. First, the learning rate (α) shows the speed at which errors are reduced over the number of occurrences of a particular

skill or concept. The slope of the graph at various points depicts the learning rate at that point. Second, χ describes the students' performance on the first trial, and therefore shows how difficult the students found the particular domain or problem set; the higher the number, the more difficult. Therefore, if the domain or problem set was kept similar and yet the χ values differed between groups of students, it could be argued that the group with the higher initial error rate (i.e. the group that performed better on the first trial and therefore found the domain easier than the other groups) must have had prior domain knowledge or higher expertise than the other groups. Third, the fit of the graph (R^2) illustrates the variability of the data within that group. The fit shows how well students used the skill they learned previously i.e. transferability of the skill learned. The better the fit, the higher the transferability. And finally, learning curves gradually decrease in slope over time i.e. the learning rate reduces. The value of the bottom of the curve shows the probability of the student making an error in subsequent occurrences of the same concept i.e. how well the student has learned the concept. If the learning rate has decreased to near-zero, and the probability of making errors is still quite high (the bottom of the curve is high), then the student's learning can be considered shallow. The lower the bottom of the curve, the deeper the learning.

4 The Data and the Methods Used

The main dataset (named dataset A) for this research was taken from an online version of SQL-Tutor that is available to students from around the world who were given free access when they bought certain SQL text books. Data for any student that made less than five attempts was excluded. The remaining data consisted of 1803 users who made a total of 100,781 attempts and spent just over 1,959 active hours on the system. Active hours is time spent actively solving the problem; not just session times. Students in this dataset on average solved 70% of the problems attempted; giving a grand combined total of 19,604 solved problems.

We extracted the number of submissions for each student from the individual student logs. Each submission was then categorised as either a valid attempt or a request for help (RFH).A valid attempt occurs whenever a student submits a solution that is different to their previous solution. An attempt need not be correct to be valid. When a student submits either an empty solution or the same solution twice in a row, this is interpreted as a RFH. Here, the student is hoping that more hints will be given on each submission. SQL-Tutor automatically increments the help level to a maximum of 3. Equation 2 shows the relationship between submissions, attempts, and RFH.

$$\text{Submissions} = \text{Attempts} + \text{Requests for help} \tag{2}$$

We further categorised each valid attempt into two categories: high-level help (HLH) attempts or low-level help (LLH) attempts, and deduced the total number of HLH attempts for each student. To normalise the HLH attempts value over all

the students, we calculated an HLH ratio; $0 \leq HLH \leq 1$. (See equation 3). This ratio categorises students' HLH seeking behaviour by showing us how frequently a student uses high-level help; low frequency HLH students are those that used high-level help very infrequently and vice versa. For example, a student with an HLH ratio of 1 uses HLH on every attempt, whereas a student with HLH ratio of 0.1 uses HLH once every ten attempts. Students were then ordered according to their HLH ratio; from low frequency HLH users to high frequency HLH users.

$$\text{HLH ratio} = \frac{\text{Number of high-level help attempts}}{\text{Total number of attempts}} \tag{3}$$

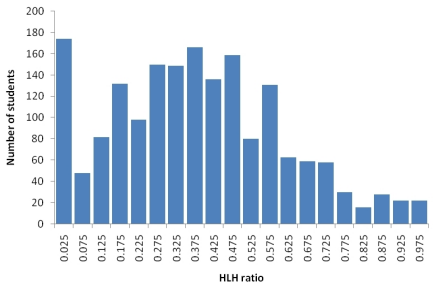

Fig. 2. Frequency distribution of users in the various HLH user groups in dataset A.

A frequency graph of students and their HLH ratio is shown in Fig. 2. We then divided the population into ten groups (A1-A10) depending on their HLH ratio; the groups were 0.1 HLH apart. The logs and student models were analysed for each group, and learning curves were plotted.

Each problem in SQL-Tutor is assigned a difficulty level by the teacher. Difficulty levels range from 1 (easiest) to 9 (most difficult). For each student, we mined the difficulty levels of problems attempted and recorded the *maximum difficulty level of problem solved* (MDLS). In our initial analysis, we used the MDLS to see how students from each group were able to learn skills and progress through to more difficult problems. Manual analysis of the logs showed that for students in the higher HLH groups, there was a large difference between the difficulty level of problems attempted and those solved; they attempted much more difficult problems than they solved. There also seemed to be a high number of abandoned problems after valid attempts were made. Furthermore, in many cases the initial (incorrect) solutions were very different to the ideal solution. It was as though the high HLH students were employing a "guess then copy" method to solving problems. In contrast, the lower HLH students created answers that were usually more similar to a correct solution, then with each attempt submitted closer approximations to a correct solution until the problem was solved. Due to this, another metric, termed the *maximum difficulty level of problem solved without using HLH* (MDLS-WH) was also calculated and used.

The results are summarised in the next section.

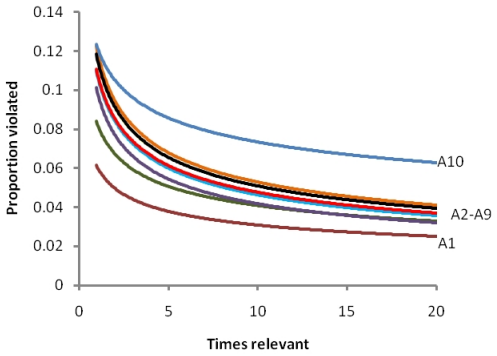

Fig. 3. The learning curves for the ten HLH user groups (A1-A10).

5 Results and Discussion

5.1 Frequency Distribution of Users According to Their HLH

The frequency distribution graph in Fig. 2 shows the number of users grouped according to their HLH. This graph has an interesting shape. From $HLH > 0.05$, a positive skewed normal curve exists. There is also a peak external to the normal distribution in the first section of the graph: $HLH \leq 0.05$; these are students that are very low users of HLH. Manual inspections of the logs seem to indicate the presence of at least two distinct types of students within this first section ($HLH \leq 0.05$): one with higher domain expertise and the other with low domain expertise. This section of students needs to be analysed further. It is understandable that the students with high expertise are low users of HLH. However, the students with low expertise who do not use HLH could be those who either have low meta-cognitive (help-seeking) skills or, due to social norms, feel that looking at HLH constitutes a form of "cheating". For comparison, frequency graphs were also plotted for two other smaller sets of data, and we found that the results were very similar. This shows that this trend is persistent across populations, and requires deeper analysis.

5.2 Learning Curves for Each HLH User Group

Learning curves were calculated and drawn for each of the ten groups (A1-A10). Fig. 3 shows the learning curves for all the groups plotted on the same graph for ease of comparison. To avoid clutter, the equations have been excluded from the graph but are listed in table 1.

Several points can be noted from the learning curves:

Table 1. Power curve equations and fits (R^2) for the ten HLH groups (A1-A10).

Group	HLH ratio	Power curve equation	R^2 (Fit)
A1	0.0 - 0.1	$y = 0.061x^{-0.30}$	0.844
A2	0.1 - 0.2	$y = 0.084x^{-0.31}$	0.956
A3	0.2 - 0.3	$y = 0.101x^{-0.38}$	0.955
A4	0.3 - 0.4	$y = 0.109x^{-0.37}$	0.956
A5	0.4 - 0.5	$y = 0.110x^{-0.36}$	0.965
A6	0.5 - 0.6	$y = 0.123x^{-0.39}$	0.961
A7	0.6 - 0.7	$y = 0.122x^{-0.36}$	0.953
A8	0.7 - 0.8	$y = 0.115x^{-0.35}$	0.953
A9	0.8 - 0.8	$y = 0.118x^{-0.36}$	0.912
A10	0.9 - 1.0	$y = 0.123x^{-0.22}$	0.956

1. The curves follow the same approximate order of the HLH use. The higher the HLH use, the shallower the learning. For example, students in the group A10 (those with the highest HLH) portray shallow learning; even when their learning rate approaches zero, their probability of making errors is still comparatively high.
2. The fit for power curve is very high across all groups. The lowest fit for curve is in A1 ($R^2 = 0.844$). This could be because, as proposed earlier, A1 contains at least two distinct sub-groups of students: the students with high expertise, and the students with low expertise who persevere with problem solving without utilising HLH. The transferability of skills could vary within these sub-groups, reducing the overall fit.
3. Since learning curves A2-A9 are very similar, this leaves us with three distinct categories of learning curves: A1, A2-A9, and A10. Fig. 4 shows the three learning curves plotted on a single graph. The middle range of HLH users displays similar learning (i.e. depth, learning rate, etc.), while the extreme high and low HLH users display markedly different learning.
4. The χ value increases as HLH use increases. This means that high HLH students find the domain more difficult than low HLH students. There is a marked difference between the χ value of A1 and the other groups. This could also show that A1 contains experts (or at least students with prior domain knowledge).
5. Curves A2-A9 report similarly high learning rates. A10, the highest HLH has the lowest learning rate (0.22). This could be because students that use HLH extremely frequently do not actively engage in the material, think for themselves, utilise their meta-cognitive skills (such as reflecting or explaining their actions), or participate in the benefits of deliberate practice (e.g. learning from errors). This causes much lower learning rates. The group of users that use HLH the least (i.e. group A1) has the second lowest learning rate. This could be because both the sub-groups (experts and low expertise students) in A1 both have low learning rates. The experts in A1 already find the domain easy, thus starting with low error rates (i.e. χ is low), and therefore reach their asymptote of learning very quickly. The novices in A1 who

persevere with problem solving without utilising HLH also have low learning rates. In this case, the ITS is therefore most beneficial for the majority of students who are in the middle HLH range, producing relatively similar, high learning rates.

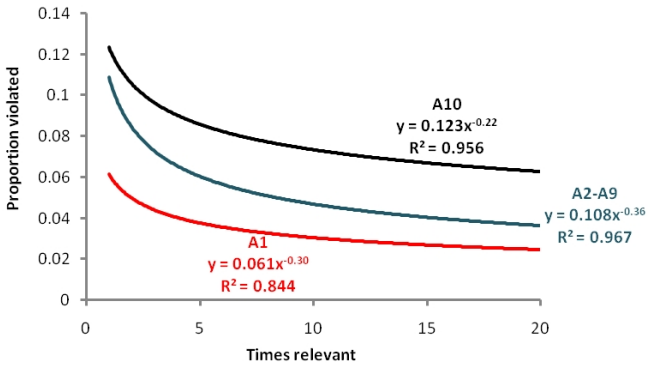

Fig. 4. The learning curves for the HLH user groups. In this graph (unlike Fig. 3), groups A2-A9 are combined. This was done as the curves for A2-A9 were very similar in nature.

5.3 Maximum Difficulty Level of Problems Solved by Students

The Zone of Proximal Development (ZPD) [7] is the area just beyond the student's ability or more specifically, it is the difference between what a learner can do without help and what they cannot do without help. In ITS terms, the problems found within the ZPD are just challenging enough for the student to solve with some help (i.e. LLH and moderate amounts of HLH) from the tutor. It is also said to be the area in which the highest rate of learning occurs. Like most ITSs, SQL-Tutor attempts to keep the student within their ZPD by guiding them through increasing levels of difficulty, while providing LLH, as their expertise in the domain increases; the student is also free to use HLH as necessary.

As mentioned earlier, problems in SQL-Tutor range in difficulty level from 1 to 9. Although the range (1-9) seems small, there is quite a difference in difficulty between levels, such that the difficulty (and type of concepts covered) between a problem of level 1 and another of level 5, for example, is considerable.

As discussed in the section above, we initially collected the *maximum difficulty level of problems solved* (MDLS); this included problems solved irrespective of the level of help used. Another metric, *the maximum difficulty level of problems solved without HLH* (MDLS-WH) was also ascertained for each user from their logs. We determined that the MDLS-WH would give an indication of how far the user progressed through the domain on their own, and thus give us a

clue, albeit a vague one, of the student's expertise in the domain. We also collected the maximum difficulty level of problems attempted (MDLA) for each student. The average MDLA, MDLS, and MDLS-WH for each HLH user group was calculated and plotted, and is shown in Fig. 5.

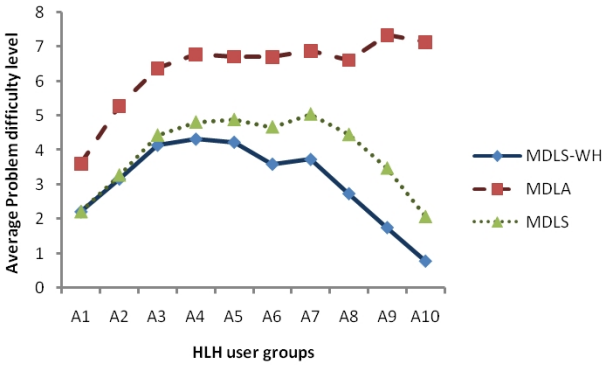

Fig. 5. The average maximum difficulty of problems attempted (MDLA), solved (MDLS), and solved without using HLH (MDLS-WH) for students in the HLH groups. Note the difference between the MDLA and MDLS-WH as HLH use increases.

As shown by the graph, the MDLA is always higher than the maximum difficulty of solved problems irrespective of any help levels. This is expected. The MDLS-WH increases steadily as HLH use increases, peaking at the A4 group (MDLS-WH = 4.32). After this, as HLH use increases, MDLS-WH decreases. The high HLH groups had very low MDLS-WH values (e.g. average MDL for A9 = 1.75) with the lowest MDLS-WH average occurring in the highest HLH user group (A10). Although the minimum problem difficulty level in SQL-Tutor is 1, the average MDLS-WH in A10 was 0.77. This was due to a number of students not solving any problems on their own, without HLH. The lowest HLH user group (A1) also reported a low MDLS-WH average (2.21). This is similar to the trend found in the learning curves above.

What is most interesting about this graph is the difference between the MDLA and MDLS-WH. The higher the HLH use, the greater the difference between the two graphs.

This shows that students who use low to moderate amounts of HLH, progress through the problem set, solving more difficult problems on their own compared to students who are either high or low frequency users of HLH. Students who are high HLH users attempt increasingly difficult problems, much harder problems on average than any other group of students. However, these students only solve very easy problems (either on their own, or using HLH), choosing to abandon problems after viewing the HLH. It is as if they have a very low expertise level, but choose to attempt problems far beyond their ZPD.

6 Future Work

Immediate future work for the authors include deeper analysis of certain groups more thoroughly (for example the composite group A1). In this paper, we divided the entire sample (evenly by HLH ratio) into ten groups. Another method would be to divide the sample into two separate groups first before beginning analysis. This first split would be dependent on the shape of the frequency distribution graph (Fig. 2), i.e. students who fall in the normal distribution of the graph ($HLH > 0.05$), and students outside the normal distribution (the group with $HLH < 0.05$). This could be interesting as this frequency trend is common across the three separate samples that we analysed. We also would like to attempt to ascertain if students remain in one group or migrate between groups during the duration of their learning.

This research forms a part of the larger observational analysis work done on learning behaviour by various researchers. These types of analysis can then be used to form a basis to categorise students, as they work on an ITS, into various groups depending on their help-seeking behaviour. Students in each particular group can then receive customised pedagogical and intervention strategies, appropriate to the group to which they belong.

One of the challenges in creating systems that attempt to increase the effectiveness of learning is observing and comprehending the behaviour displayed by various groups of students in particular contexts. This research examines students' behaviour with regards to seeking high-level help. It forms a piece in the larger set of observations gathered by researchers, which then gives us some basis for creating customised pedagogical strategies.

References

1. Mitrović, T.: An Intelligent SQL Tutor on the Web. IJAIED. 13, 173–197 (2003)
2. Aleven, V., Koedinger, K.: Limitations of Student Control: Do Students Know when They Need Help? In: ITS 2000, pp. 292–303. LNCS 1839, Berlin(2000)
3. Baker, R., Corbett, A., Koedinger, K., Evenson, S., Roll, I., Wagner, A., Naim, M., Raspat, J., Baker, D., Beck, J.: Adapting to When Students Game an Intelligent Tutoring System. In: ITS 2006, pp. 392–401. Jhongli, Taiwan (2006)
4. Baker, R., Corbett, A., Koedinger, K., Roll, I.: Detecting When Students Game the System, Across Tutor Subjects and Classroom Cohorts. In: UM 2005, pp.220–224. LNAI 3538, Springer-Verlag Berlin Heidelberg (2005)
5. Koedinger, K., Mathan, S.: Distinguishing Qualitatively Different Kinds of Learning Using Log Files and Learning Curves. In: ITS 2004 Log Analysis Workshop., pp. 39–46. Maceio, Brazil (2004)
6. Martin, B., Koedinger, K., Mitrovic, T., Mathan, S.: On Using Learning Curves to Evaluate ITS. In: AIED 2005, pp. 419–426. IOS Press, Amsterdam (2005)
7. Vygotsky, L.: Mind in Society: The Development of Higher Psychological Processes. Harvard University Press, Cambridge, MA (1978)

Analysing High-Level Help-Seeking Behaviour in ITSs

Moffat Mathews, Tanja Mitrović, and David Thomson

Computer Science & Software Engineering,
University of Canterbury, New Zealand
{moffat,tanja,djt88}@cosc.canterbury.ac.nz

Abstract. In this paper, we look at initial results of data mining students' help-seeking behaviour in two ITSs: SQL-Tutor and EER-Tutor. We categorised help given by these tutors into high-level (HLH) and low-level help (LLH), depending on the amount of help given. Each student was grouped into one of ten groups based on the frequency with which they used HLH. Learning curves were then plotted for each group. We asked the question, *"Does a student's help-seeking behaviour (especially the frequency with which they use HLH) affect learning?"* We noticed similarities between results for both tutors. Students who were very frequent users of HLH showed the lowest learning, both in learning rates and depth of knowledge. Students who were low to medium users of HLH showed the highest learning rates. Least frequent users of HLH had lower learning rates but showed higher depth of knowledge and a lower initial error rate, suggesting higher initial expertise. These initial results could suggest favouring pedagogical strategies that provide low to medium HLH to certain students.

A primary aspect of researching and developing adaptive systems is to try and understand the behaviour of those using the system. Being able to comprehend various types of behaviour gives us the basis to form strategies to adequately, effectively, and even adaptively aid users of the system. This is particularly the case in Intelligent Tutoring Systems (ITSs), where understanding each student's behaviour is critical to creating and implementing suitable pedagogical strategies to appropriately guide each student adaptively through their learning tasks in order to maximise their learning. One such type of behaviour is the way in which a student requests and utilises help. Help-seeking behaviour has been studied in various contexts; from traditional teaching methods in the classroom to e-learning applications. It has long been noted in education literature that seeking help and way in which it is sought affects learning [1]. Certain aspects of help-seeking behaviour (such as gaming [2]) have been researched in the context of ITSs [3]. In an adaptive system, one method of studying users' behaviour is by mining data collected from users (e.g. user models and logs).

In this paper, we discuss the initial results of data mining student logs and user models for help-seeking behaviour in two ITSs, namely SQL-Tutor [4] and EER-Tutor [5]. We grouped students by the frequency with which they used help

W. Nejdl et al. (Eds.): AH 2008, LNCS 5149, pp. 312–315, 2008.

and tried to determine if there were differences in learning between the groups of students. We did this by plotting learning curves for each group and seeing if any trends existed, and if these trends were similar between the two ITSs.

SQL-Tutor is a constraint-based modelling (CBM) ITS that provides intelligent and adaptive guidance in the domain of SQL database querying. SQL-Tutor has been used since 1998 in tertiary undergraduate database courses. The student spends the majority of time solving problems in the task environment. The task environment contains the problem text, solution workspace, feedback pane, and problem context information (e.g. information about the schema). On submission of their solution, a student can receive help from six levels of problem-related feedback. These levels increase in the amount of help, and are 1. Simple Feedback, 2. Error Flag, 3. Hint, 4. Partial Solution, 5. List All Errors, and 6. Complete Solution. On each incorrect submission, the help level automatically increments to a maximum of 3 (i.e. hint). Help levels are selected via a combo box and the student has the ability to override the current selection at any time by selecting a different level. We divided the help into two categories depending on the amount of help given: low-level help (LLH) for the first three help levels and high-level help (HLH) for levels four, five, and six. Furthermore, LLH automatically increments on incorrect submissions whereas HLH has to be selected by the student.

EER-Tutor (Enhanced-Entity Relationship Tutor) is a CBM ITS that teaches conceptual database design using the Enhanced Entity Relationship Model, and provides students with problems to practise their entity relationship modelling skills in a coached environment. Developed initially as KERMIT (Knowledge-based Entity Relationship Modelling Intelligent Tutor) then ER-Tutor (Entity-Relationship Tutor) and now EER-Tutor, this ITS has also had many years of successful use with students in tertiary undergraduate database courses. The help-levels in EER-Tutor are similar to SQL-Tutor and thus make it easy for comparison. As with SQL-Tutor, we divided help into LLH and HLH.

Although these two ITSs deal with database related areas, each domain is very different, Furthermore, the method of solving problems (even to the point of *text* versus *diagrammatic)* is considerably different.

1 Method

In both datasets, data for students who made less than five attempts was omitted from the analysis. The SQL-Tutor dataset consisted of 1,803 students who made a total of 100,781 attempts, and spent just over a total of 1,959 active hours on the system. EER-Tutor dataset consisted of 936 students who made a total of 43,485 attempts, and spent just over 2,830 active hours on the system.

To enable us to compare the frequency of HLH use among students, we calculated an HLH-Ratio ($\frac{\text{Number of HLH attempts}}{\text{total number of attempts}}$) for each individual. For example, a student with an HLH ratio of one used HLH on every attempt; in contrast a student with an HLH ratio of zero never used HLH. For comparison between groups of users with similar HLH, ten groups ($A1 - A10$) were formed, each with an HLH ratio range of 0.1. Students were placed into groups depending on their

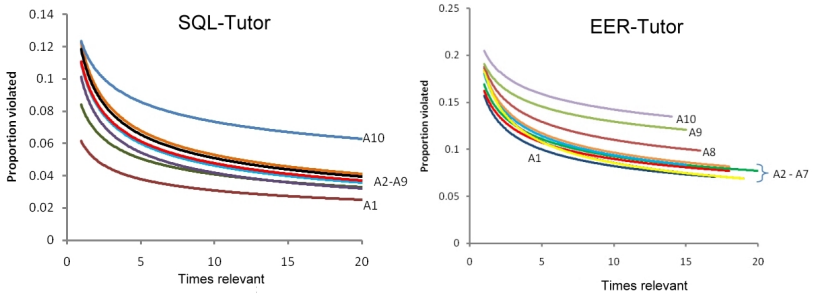

Fig. 1. Learning curves for the HLH groups (A1-A10) for SQL-Tutor and EER-Tutor

HLH ratio, such that students in group A10 (who used HLH 90–100% of the time) were the most frequent users of HLH while the least frequent HLH users were in group $A1$. Learning curves were then plotted for each group (Figure 1).

Table 1. Power curve equations and fits (R^2) for the ten HLH groups ($A1 - A10$) in SQL-Tutor and EER-Tutor

		SQL-Tutor			EER-Tutor		
Group	HLH ratio	Users	Curve equation	R^2 (Fit)	Users	Curve equation	R^2 (Fit)
A1	0.0 - 0.1	222	$y = 0.061x^{-0.30}$	0.844	210	$y = 0.156x^{-0.28}$	0.959
A2	0.1 - 0.2	214	$y = 0.084x^{-0.31}$	0.956	186	$y = 0.162x^{-0.25}$	0.963
A3	0.2 - 0.3	248	$y = 0.101x^{-0.38}$	0.955	120	$y = 0.169x^{-0.26}$	0.963
A4	0.3 - 0.4	315	$y = 0.109x^{-0.37}$	0.956	103	$y = 0.185x^{-0.33}$	0.938
A5	0.4 - 0.5	295	$y = 0.110x^{-0.36}$	0.965	89	$y = 0.181x^{-0.28}$	0.967
A6	0.5 - 0.6	211	$y = 0.123x^{-0.39}$	0.961	57	$y = 0.183x^{-0.28}$	0.947
A7	0.6 - 0.7	122	$y = 0.122x^{-0.36}$	0.953	51	$y = 0.184x^{-0.33}$	0.978
A8	0.7 - 0.8	88	$y = 0.115x^{-0.35}$	0.953	51	$y = 0.189x^{-0.23}$	0.954
A9	0.8 - 0.8	44	$y = 0.118x^{-0.36}$	0.912	34	$y = 0.190x^{-0.16}$	0.858
A10	0.9 - 1.0	44	$y = 0.123x^{-0.22}$	0.956	35	$y = 0.204x^{-0.15}$	0.878

2 Results and Discussion

The power curve equations and fits are shown in Table 1. The results discussed are similar for both tutors.

All learning curves have a very good fit (R^2), with the lowest fit being just 0.844. The degree of fit usually indicates level of transferability of the skills that were learned. For example, a low fit indicates high variability in the error rates (i.e. high deviation of points from the power curve) indicating that error rates still vary each time a particular concept is encountered (i.e. low transferability). This result indicates that whatever skills students are learning are also transferable. This does not indicate that all students are learning the same skills.

The exponent in the equation indicates the learning rate. As can be seen from Table 1, the learning rates are highest for students who are low to medium users of HLH. Students that are extremely high users of HLH (e.g. A10) have the lowest learning rates. These students also display shallow learning. This can be seen from the point at which the slope of the learning curves approximates zero. For extremely high HLH users, this point still shows a high error rate, indicating that the concept has not been learned to any great depth. This could be because students that rely heavily on HLH do not actively think for themselves or engage in deliberate practice, and therefore do not get the opportunity to learn from their mistakes.

The coefficient of x (known as χ) shows the initial error rate. Low χ usually means the presence of expertise or previous experience and vice-versa. The χ value for group $A1$ in SQL-Tutor shows this expertise or prior knowledge. Manual inspection of logs indicated the presence of students with higher expertise in this group. As a consequence, students who have a higher χ find the domain more difficult than those with lower χ values. From Figure 1, we can see that the students who used the least help had the least χ, whereas students who used the most help had the highest χ and therefore found the domain more difficult.

Although these initial results provide a good basis for understanding one aspect of help-seeking behaviour, and thus aids in creating pedagogical strategies, it cannot be construed from these results that providing low to medium help to students will automatically increase learning. Other factors such as the meta-cognitive ability (e.g. help-seeking skills) of students, their upbringing, and even their cultural influences also need to be considered. It could also be that students who are slower to learn are less confident and therefore seek HLH more often.

In the near future, we intend to analyse the effect of other variables such as time spent on attempts, number of problems solved, and difficulty of problems solved on these groups of students.

References

1. Gall, S.N.: Help-Seeking Behaviour in Learning. Review of Research in Education 12, 55–90 (1985)
2. Baker, R., Corbett, A., Koedinger, K., Roll, I.: Detecting When Students Game the System, Across Tutor Subjects and Classroom Cohorts. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 220–224. Springer, Heidelberg (2005)
3. Aleven, V., Koedinger, K.: Limitations of Student Control: Do Students Know when They Need Help? In: Gauthier, G., VanLehn, K., Frasson, C. (eds.) ITS 2000. LNCS, vol. 1839, pp. 292–303. Springer, Berlin (2000)
4. Mitrović, T.: An Intelligent SQL Tutor on the Web. IJAIED 13, 173–197 (2003)
5. Suraweera, P., Mitrović, T.: An Intelligent Tutoring System for Entity Relationship Modelling. JIJAIED 14, 375–417 (2004)

Kong, S.C., Ogata, H., Arnseth, H.C., Chan, C.K.K., Hirashima, T., Klett, F., Lee, J.H.M., Liu, C.C., Looi, C.K., Milrad, M., Mitrovic, A., Nakabayashi, K., Wong, S.L., Yang, S.J.H. (eds.) (2009). Proceedings of the 17th International Conference on Computers in Education [CDROM]. Hong Kong: Asia-Pacific Society for Computers in Education.

Does Framing a Problem-Solving Scenario Influence Learning?

Moffat MATHEWS, Antonija MITROVIC
Intelligent Computer Tutoring Group, University of Canterbury, New Zealand
{moffat, tanja}@canterbury.ac.nz

Abstract: In this paper, we discuss potential effects of framing, a pedagogical strategy used by some teachers, with the view of implementing it in an intelligent tutoring system. The process of framing a learning activity, in our case problem solving, consists of having the activity in between a pre-action (or priming) phase and a post-action (or reflective) phase. We also describe an evaluation study where the experimental and control groups participated in a framed and non-framed learning session respectively. The pre- and post-action phases were whiteboard group sessions led by an SQL domain expert. The problem-solving phase was done solely on SQL-Tutor, an ITS for database querying. During the problem-solving phase, the experimental group solved the same number of problems in 23% less time than their counterparts in the control group. The type of problems solved and the high-level help used were similar for both groups. Although the learning gains were high, they were similar across groups, questioning the effectiveness of group-based reflection. The experimental group was also more efficient, expending less effort for similar gains. After examining various learning theories and analyzing the study results, we conclude that this is a valid teaching format for the next step of our research: investigating its implementation and evaluation within an ITS

Keywords: ITS, Teaching models, Framing learning sessions, Evaluation

Introduction

Intelligent Tutoring Systems (ITSs) today are generally problem-solving environments where students can practice what they have learned. ITSs are usually not used as the sole source of students' learning, but rather used in conjunction with other forms of declarative knowledge transfer (such as lectures) to complement the learning process. Research such as this, which looks at altering one or more factors of the learning scenario to maximize some aspect of the student's learning, is commonly done in the field of Artificial Intelligence in Education.

To this end, researchers work on creating, modifying, and evaluating various theories. Teachers implement various combinations of theories in the form of activities or strategies, by adapting them to fit the particular domain, set of students, and curriculum. One problem that educators face when implementing a new strategy is the difficulty in evaluating its effect on the various aspects of learning. The purpose of this paper is to discuss one such adaptation, namely *framing the learning activity*. We discuss the relevant theories and teaching methods in Section 1. Section 2 present the context and design of our study conducted at the University of Canterbury in 2008. We present our hypotheses about the effects of framing in Section 3, and the results of the study in Section 4. Section 5 discusses the results in the light of our hypotheses, followed by conclusions and avenues for further

research in Section 6. We also aim to show that beyond simple pre- and post-tests, such a setup (with ITSs) can be used to evaluate new strategies.

1. Related Work

A learning session is *framed* when the learning activity is immediately preceded by a pre-action (or *priming*) phase and followed by a post-action (or *reflection*) phase. In our case, the learning activity is solving problems within an ITS. In the pre-action phase, the teacher (re-) introduces the target concepts (concepts that would be used in the problem-solving phase), links them to previously learned concepts (explains where it fits in the domain), works through examples, discusses common misconceptions, and sets the "boundaries" for the session. The pre-action phase is not the same as a traditional lecture, either in its intended purpose or in its content. In the problem-solving phase, students solve problems relating to the target concepts. In the post-action phase, the teacher prompts each student to reflect on his or her problem-solving experiences. Students are encouraged to analyse their errors (including the source of these errors) thereby uncovering misconceptions. The common mistakes made during the problem-solving session are usually worked through as a group.

There are several theories that make framing a plausible teaching strategy. Cognitive Load Theory [7] suggests that problem solving for novices generates heavy working memory loads, which could be detrimental to learning. When a student is presented with new items to learn, the working memory limit comes into play. To balance these loads, teachers should provide guided instruction, help narrow the problem search space by creating “boundaries” to each session, and alleviate the working memory restriction by making sure that only items relevant to the task are loaded into the working memory. This is exactly what happens during the priming phase.

Meta communication about the presentation of the subject is important with respect to learning [9]. Prior to the learning activity, the student needs to know the boundaries of the lesson segment (exactly what the lesson will contain), which should be well defined by the teacher. The student needs to know the content of the session, differentiating between the old and the new material. They also need to know the links between the new knowledge and previously learned knowledge [9].

Many learning models view learning as a cyclic process, around which knowledge acquisition, knowledge application (including experimentation and experience), and reflection (or knowledge processing) occur. Andreasen and Wu [1] discuss a few of the commonly used experiential models. Framing is a simplified (and thus possibly easier-to-implement) version of many of the models.

Reflection promotes deep learning [4, 5, 8]. Critically analysing the learning experience helps challenge the student’s underlying perception of the domain or topic, identify and correct misconceptions, and integrate new knowledge with existing knowledge [2]. This also better allows the student to transfer the newly acquired knowledge to other types of problems or scenarios. Self-explaining one’s actions [3] and monitoring one’s progress via open student models [6, 13] have been shown to be useful reflective tools that benefit learning.

2. Study Design

SQL-Tutor [11], an ITS that teaches database querying, was used in the problem-solving phase of the study. SQL-Tutor has been used by many students around the world and is used

in database courses including a second year database course (COSC226) at the University of Canterbury. COSC226 students are expected to attend lectures and a two-hour lab session (where they can practice what they learned in lectures) each week.

Thirty-eight students from COSC226 participated in the evaluation for no monetary reward. We divided them randomly into two groups: experimental and control. The idea was to perform the evaluation in a setting that was as close to the normal learning environment faced by students in COSC226 and similar courses. As such, the experimental and control sessions were held during regular course lab sessions (100 minutes long) on 14 and 15 May 2008 respectively. The students participated in the study during the lab session they normally attended throughout the course. None of the students had used SQL-Tutor previously, although most had used another ITS (EER-Tutor) in previous labs.

A set of target SQL concepts, namely the concepts covered by queries using the *Group By* and *Having* clauses were chosen for the study. Students generally find these types of queries difficult as they add another level of complexity to the *Select* statement i.e. students not only have to think in terms of tuples (rows) but also in terms of grouped tuples. Furthermore, conditions cannot only be set on tuples (using the *Where* clause) but also on groups (using the *Having* clause), which is another common point of confusion amongst novices. The ability to use aggregate functions (functions on a group, such as *sum*, *average*, *count* etc.) also adds to the complexity. For this evaluation, SQL-Tutor was restricted to only present problems relating to these target concepts. The study was held immediately after the relevant concepts had been covered in lectures.

Both groups were informed that their performance in the session did not count towards course assessment. The experimental and control sessions had five and three phases respectively within the set 100 minutes. Students in both groups completed an identical pretest in the first phase and an identical posttest in the last phase. All tests were of comparable difficulty and contained three questions relating to the target concepts. Students were required to formulate an SQL query for each question. An SQL domain expert created and marked all tests. The three questions varied in difficulty and marks were given for correct usage of SQL concepts i.e. partially correct answers received some marks. The maximum score for each test was 12, with questions being 3, 4, and 5 marks respectively.

After the pretest (first phase), the experimental group went through the pre-action (second) phase prior to the problem-solving (third) phase, which was followed by the post-action/reflection (fourth) phase and the posttest (fifth phase). The pre-action and post-action phases were ten-minute interactive, whiteboard sessions led by an SQL domain expert. Both these phases *framed* the problem-solving phase. In the pre-action phase, the expert briefly reminded students of the target concepts (which were taught in lectures previously) and, eliciting student participation, worked through a few examples. The examples, although different to those encountered in the problem-solving phase, were still of varying difficulty, covering the target concepts. The expert also discussed typical misconceptions. In the post-action phase, the concepts were reiterated. Students were prompted to reflect on their learning experience by commenting on some of their own mistakes and misconceptions. The expert also showed them the most common mistakes that are usually made during the problem-solving phase. Students were asked to find the errors (in terms of concepts and methods) in those incorrect solutions before collectively working through to reach a correct solution.

In contrast, the control group entered the problem-solving (second) phase following the pretest. They were told that they could practice problems from SQL-Tutor to improve their database querying skills. The pretest, posttest, and problem-solving phases for both groups were identical.

During the problem-solving phase, the students in both groups worked solely on SQL-Tutor without any assistance from human tutors. SQL-Tutor has its own

problem-selection strategy. When a student requests a new problem, it presents the student with a list of potential problems within the student-selected database, highlighting one of the problems as a system choice or recommendation. This choice is the system's recommendation of the *next best problem* and takes into account a variety of factors such as the student's knowledge of a particular concept, their problem-solving history (student model), and attributes of the problem (such as its difficulty level). The student can override the system's recommendation and choose a different problem. We used this problem selection strategy with the restricted set of problems for both groups.

3. Hypotheses

We had two hypotheses for this study.

Hypothesis 1: The experimental group should have a faster *speed of solving problems* than the control group. We postulated that this would be due to the pre-action phase. This is mainly because this phase balances the cognitive load on the student and primes them for the problem-solving phase. Informing the student of the target concepts, reminding them of the situations in which they are applied, and addressing potential misconceptions should mentally prepare the student for the problem-solving phase. Working through examples with the expert should arm the students with the required mental tools at the crucial time and load the necessary information into their restricted working memory to solve the problems and correct their errors.

Hypothesis 2: The experimental group should have higher gains in the tests than the control group. We postulated that this increase would be due to both pre- and post-action phases. The post-action phase involves reflection, which promotes deeper learning. Asking the student to look back and think about their problem-solving activity, including critically analyzing their mistakes gives the student a deeper understanding of the learned concepts, as shown in previous studies [4, 5, 8]. The pre-action phase would enable students to solve problems faster, and therefore achieve more during the problem-solving phase, as well as enable them to focus on important domain concept. If the experimental group solved more problems than the control group, the additional gain would merely be a function of the amount of practice they received. In addition, the experimental group might eventually attempt and solve more difficult problems than the control group, leading to deeper learning and thus a higher posttest score. However, this side effect should be minimized as the experimental group was given less overall time for the problem-solving phase, due to the fact that the overall session length was fixed (100 minutes) and the pre- and post-action phases were 10 minutes each.

4. Results

One student did not make any attempts on SQL-Tutor and was excluded from the study. The data we collected and analyzed included the pre/posttest results and just over 43 hours (total) of SQL-Tutor student models and logs in which 37 students collectively made 2,275 submissions to the system. There were 20 students in the control group and 17 in the experimental.

All students sat the pretest. There were no significant differences between the pretest scores for both groups suggesting that both groups had similar SQL knowledge prior to the evaluation. Table 1 shows the calculated means and standard deviations for both tests and the percentage gain. Not all students completed the posttest, and therefore Table 1 gives the results only for those students who took both tests. Both groups improved significantly

(p<.01) from pre- to posttest. Although the average gain for the experimental group was higher than that of the control group, the difference between groups was not significant. Further analysis of each of the three problems in the pre and posttest found that there were no significant differences between the scores of each problem between the groups.

Table 1: Matched means and standard deviations for test scores (%) and gains.

	Pretest	**Posttest**	**Gain**
Experimental group (n=11)	40.2% ($s.d$ = 29.1)	78% ($s.d$ = 17.6)	37.9% ($s.d$ = 31.9)
Control group (n=12)	47.9% ($s.d$ = 29.3)	78.5% ($s.d$ = 13)	30.6% ($s.d$ = 19.6)

The rest of the analyses were done on all 37 students. The average time spent solving problems (given in Table 2) on SQL-Tutor by students in the experimental group was 60.73 min ($s.d$=14.67) while those in the control group spent on average 79.34 min ($s.d$=33.4). This means that the experimental group spent less time solving problems, $t(27)=2.25$ $p=.016$. This in itself is not an interesting factor as the overall problem-solving phase for the experimental group was shorter than that of the control group. However, both groups attempted and solved a similar number of problems. The experimental group solved an average of 13($s.d$=8) problems each, while the control group solved on average 12.15 ($s.d$=7.8) problems each. This means that the experimental group solved problems faster than their peers in the control group ($t(26)=1.8$, $p=.03$ one-tailed, assuming unequal variances). This gave an effect size (Cohen's d) of 0.66 (medium) using the pooled standard deviation.

The number of problems solved was strongly correlated to the number of distinct constraints (or domain principles) used by the student, 0.87 for the control group and 0.8 for the experimental group.

The experimental group solved problems faster, but were the problems solved by both groups similar? For example, did the experimental group solve easier problems? Each problem in SQL-Tutor is assigned a difficulty level by the SQL domain expert who authored the problem. Difficulty levels range from 1 (easy) to 9 (difficult) with non-trivial differences in difficulty between levels. SQL domain experts have checked problem difficulty levels such that problems with the same difficulty level are of similar difficulty, even across all the databases. The calculated means and standard deviations of the difficulty levels of problems attempted and solved for both groups are shown in Table 2. There was no significant difference between the difficulty of problems attempted and solved between groups.

Table 2: Means and standard deviations for experimental and control groups.

	Experimental	**Control**
Difficulty of problems attempted	4.8 (0.29)	4.93 (0.34)
Difficulty of problems solved	4.79 (0.28)	4.83 (0.36)
Lowest difficulty of problems attempted	3.35 (0.49)	3.5 (0.51)
Highest difficulty of problems attempted	6.4 (1.1)	6.65 (1.22)
Lowest difficulty of problems solved	3.35 (0.49)	3.35 (0.51)
Highest difficulty of problems solved	6.4 (1.0)	6.4 (1.31)
Number of problems solved	13 (8)	12.15 (7.8)
Time spent on problem solving (min)	60.73 (14.67)	79.34 (33.4)
High-level Help (HLH) ratio	0.36 (0.22)	0.32 (0.23)
Request for Help (RFH) attempts	2.11 (1.36)	1.95 (1.35)
Relative learning efficiency (E)	0.31	-0.34

Another important factor that influences the number of solved problems is the amount of feedback (help) the students received. The higher problem-solving rate of the experimental group might have been the consequence of getting more help from the system. For example, did the experimental group view the full solution to problems more often than the control group? In order to analyze the effect of help, we calculated the high-level help ratio for both groups. *High-level help* (HLH) [11] is defined as the type of help given by a

system that provides (part or all of) the correct solution to the student rather than having the student to *solve* the problem; e.g. full solution is a type of HLH. Another important characteristic of HLH in SQL-Tutor is that the HLH levels have to be manually requested by the student whereas the ITS might automatically provide other types of feedback (Low-level help). The *HLH ratio* is the number of HLH attempts divided by the total number of attempts. This shows us the proportion of HLH use, from 0 (no HLH use) to 1 (the student used HLH on every attempt). The average HLH ratio for the experimental group was 0.36 (0.22) and 0.32 (0.23) for the control group; therefore on average, they requested similar amounts of HLH.

Furthermore, we calculated the *Request for Help* (RFH) [11] attempts. An RFH is an attempt where the student submits either an empty solution or the same solution as the previous attempt. As the student has not made any changes to their solution, this could be interpreted as a request for more help from the system. There were no significant differences between the RFH for both groups; on average, the experimental group made 2.11 (1.36) requests for help while the control made 1.95 (1.35).

The relative *learning efficiency* (E) is defined as the performance gained in one condition (say, the experimental condition) over the effort expended in relation to another condition (say, the control condition). A condition is more efficient if "1) their performance is higher than expected on the basis of their mental effort and/or 2) their invested mental effort is lower than might be expected on the basis of their performance" [14]. To calculate the efficiency of the problem-solving phase of each group, we used "time" as the effort spent and "test gains" as the performance measure. The relative efficiency is found by first converting each of the raw scores to a *z* score by subtracting the grand mean from the raw score and dividing by the standard deviation. E scores then are found by calculating the perpendicular distance between each *z* score and the E=0 line when plotted on a Cartesian graph. In this evaluation, the efficiency of the experimental group (E = 0.31) was significantly higher than that of the control group (E = -0.34), $t(21)=1.85$, $p=0.039$ (one-tailed, assuming unequal variances).

5. Discussion of the Results with Reference to the Hypotheses

As predicted by the first hypothesis, the experimental group had a higher problem-solving speed even though they attempted and solved problems of similar difficulty while using similar levels of help. The problem-solving speed is the number of problems a student solves per quantity of time. This should not be confused with their *learning rate.* The learning rate is the change in the probability of getting an error for a concept (the rate at which they learned the concept) per attempt. According to our hypothesis, the pre-action phase helps reduce and balance a student's cognitive load, allowing better use of their cognitive resources. It primes and readies the student, transferring the required information into their current/working memory. The boundaries set on the session reduce the student's problem search space. These tasks help the student retrieve the needed information quicker while helping the student stay on the current task of solving the problem, i.e. it makes the student faster. It does not necessarily reduce the number of attempts required for absolute novices to solve a problem, but given adequate feedback, they can link it to the information in their working memory to correct their errors faster. As these links are created more easily, this could mean that the learning rates for non-novices from the experimental group increase at a higher rate than their peers in the control group over time.

The exponential learning curves for both groups are shown in Figure 1. Both curves start from approximately the same position at attempts = 1, showing that both groups had similar initial knowledge of the concepts. The exponent shows the learning rate [10]. The R^2

value shows the fit to curve and ranges from 0 to 1. As mentioned, the learning *rates* for both groups are very similar.

Furthermore, the experimental group was significantly more efficient in their problem-solving phase than the control group. In other words, while they did not learn more than the control group, they expended significantly less effort and therefore were more efficient.

Surprisingly, the second hypothesis was not supported by this study. Both groups had significant, yet similar gains in the posttest. One reason could be that we need a longer study with more participants to get significant differences in the learning gains. However, another reason could be that reflection is a personal activity. It requires a student to actively engage in evaluating their own actions and solutions (or in some cases, other's actions). The quality of their analysis and the assessment of their actions affect the quality of the reflection and thus the outcome on the depth of learning produced by reflecting. In a group, the individual's responsibility and accountability decreases. These are both key components of reflection. Instead, they rely more heavily on others or the group leader to prompt and inform them of their actions and errors. Furthermore, not all students might have developed the skills required for adequate reflection, a meta-cognitive task. In this evaluation, we ran the reflection phase as many teachers do, in a group. Did the students truly reflect on their actions in our group reflection phase? An important question from this study could be "are group reflection exercises truly effective?" An item for future work is to investigate the

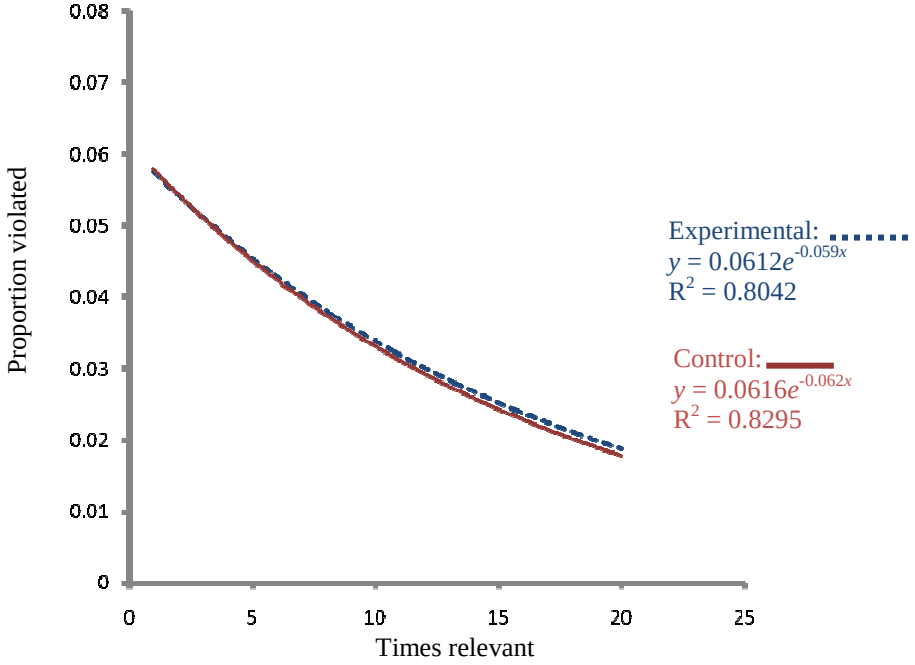

Figure 1. Learning curves for the experimental and control groups.

possibility of implementing the entire framed session in an ITS. This would also make the post-action phase a non-group activity. Evaluation of that system would give us an idea about any differences between individual reflection and group reflection scenarios.

6. Conclusions

The use of framing as a teaching strategy seems to be a valid option, both when considering the various learning theories and the evaluation results. In our study, the primary learning activity was problem solving in SQL-Tutor, a constraint based tutor that teaches queries in the SQL language. The students who had been through a priming session before attempting

problem solving were more efficient in their learning than their peers, as evidenced by a higher problem completion speed. The comparison of their learning to the control group, which did not have priming, showed that both groups of students learned the domain knowledge to the similar depth: the experimental group just did it faster. The experimental group also had a group reflection session, led by a human teacher; however, we observed no differences due to this reflection phase.

This study has also left us with a few questions. For example, how much time should be spent in each of the phases? We used 10 minutes for priming and reflection, but not because of any theory. Should the framing time perhaps be a factor of the problem-solving time? Is there an optimal time for each phase such that the overhead of the framing time does not compromise the overall time of the learning session? Another limitation of the study could be the relatively small number of participants.

In the previous section, we posed a question regarding the usefulness of group-based reflection in comparison to individual, tailored reflection. In hindsight, an additional delayed test might have also provided a little more insight into the depth and retention of learned knowledge.

We plan to enhance SQL-Tutor by providing the priming phase within the system, and conduct another evaluation study of such an enhanced version of the system in 2009.

References

[1] Andreasen, R. J., & Wu, C.-H. (1999). *Study Abroad Program as an Experiential, Capstone Course: a Proposed Model.* Paper presented at the 15th Annual Meeting of the Association for International Agricultural and Extension Education, Trinidad & Tobago.

[2] Atkins, S., & Murphy, K. (1993). Reflection: A Review of the Literature. *Journal of Advanced Nursing, 18*(8), 1188-1192.

[3] Chi, M. T. H. (2000). Self-Explaining Expository Texts: The Dual Processes of Generating Inferences and Repairing Mental Models. In R. Glasser (Ed.), *Advances in Instructional Psychology: Educational Design and Cognitive Science* (Vol. 5, pp. 161 - 238). Mahwah, NJ: Lawrence Erlbaum Associates.

[4] Katz, S., Allbritton, D., & Connelly, J. (2003). Going Beyond the Problem Given: How Human Tutors Use Post-Solution Discussions to Support Transfer. *International Journal of Artificial Intelligence in Education, 13*(1), 79-116.

[5] Katz, S., Connelly, J., & Wilson, C. (2007). Out of the Lab and into the Classroom: An Evaluation of Reflective Dialogue in Andes. In R. Luckin, K. R. Koedinger & J. E. Greer (Eds.), *Artificial Intelligence in Education: Frontiers in Artificial Intelligence and Applications* (Vol. 158, pp. 45-432): IOS Press.

[6] Kay, J. (1997). *Learner Know Thyself: Student Models to give Learner Control and Responsibility.* Paper presented at the International Conference on Computers in Education, Sarawak, Malaysia.

[7] Kirschner, P. A., Clark, R. E., & Sweller, J. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist, 41*(2), 75-86.

[8] Lee, A. Y., & Hutchison, L. (1998). Improving Learning from Examples through Reflection. *Journal of Experimental Psychology: Applied, 4*(3), 187-210.

[9] Leinhardt, G., & Ohlsson, S. (1990). Tutorials on the structure of tutoring from teachers. *Journal of Artificial Intelligence in Education, 2*(1), 21-46.

[10] Martin, B., Koedinger, K. R., Mitrovic, A., & Mathan, S. (2005). *On Using Learning Curves to Evaluate ITS.* Paper presented at the 12th International Conference on Artificial Intelligence in Education, Amsterdam.

[11] Mathews, M., & Mitrovic, A. (2008). *How does students' help-seeking behaviour affect learning?* Paper presented at the 9th International Conference on Intelligent Tutoring Systems, Montreal, Canada.

[12] Mitrovic, A. (2003). An Intelligent SQL Tutor on the Web. *International Journal of Artificial Intelligence in Education, 13*(2-4), 173-197.

[13] Mitrovic, A., & Martin, B. (2007). Evaluating the Effect of Open Student Models on Self-Assessment. *Int. J. Artif. Intell. Ed., 17*(2), 121-144.

[14] Paas, F. G. W. C., & Merrinboer, J. J. G. V. (1993). The Efficiency of Instructional Conditions: An Approach to Combine Mental Effort and Performance Measures. *Human Factors: The Journal of the Human Factors and Ergonomics Society, 35*, 737-743.

Mohd Ayub A. F. et al. (Eds.) (2011). Workshop Proceedings of the 19th International Conference on Computers in Education. ChiangMai, Thailand: Asia-Pacific Society for Computers in Education.

Incorporating Framing into SQL-Tutor

Moffat MATHEWS & Antonija MITROVIC
Intelligent Computer Tutoring Group, University of Canterbury, New Zealand
moffat.mathews@canterbury.ac.nz
tanja.mitrovic@canterbury.ac.nz

Abstract: In this paper, we present our work on framing with the view of implementing it in an Intelligent Tutoring System (ITS). The process of framing a learning activity, in our case problem solving, consists of having the activity in between a pre-action (or priming) phase and a post-action (or reflective) phase. In previous work, we found that simulated framing, in which the priming and reflection phases were led by a human teacher while the learning activity itself was performed in an ITS, significantly reduces learning time and requires less effort for similar gains. This paper presents the next stage of the project, in which the priming phase is implemented in the ITS. We performed a pilot study using the extended system, which resulted in the same trends as simulated framing.

Keywords: Intelligent tutoring systems, framing, teaching strategies

Introduction

In previous work [1] we have presented the initial results on the framing teaching strategy. Framing is a pedagogical strategy that we have distilled from educators' practices in tertiary institutions and high schools. The strategy consists of three sequential phases whereby the learning activity (action phase) is preceded by a pre-action (or priming) phase and followed by a post-action (or reflection) phase. All three phases together form a learning session. In the classroom, students generally participate in the pre- and post-action phases as a group, while the action phase is done either individually or as a group.

The purpose of the pre-action phase is to prepare the student for the learning activity by helping them focus on the concepts that will be used in the learning activity. The aim of this phase is not to teach them the declarative knowledge required but to "set the scene" for the learning activity. Teachers could lead the short, interactive session by (re-) introducing the target concepts, linking them to previously learned concepts, working through examples, discussing common misconceptions, and setting the "boundaries" for the session.

The learning activity phase immediately follows the pre-action phase. Here, the student takes part in some activity that helps them interact with material relating to the target concepts. For example, students might solve problems, engage in discussion, conduct exploratory research, or run experiments. This phase is self-directed, enabling the student to put into practice what they have learned, and the teacher might provide feedback.

Once the learning activity is complete, the teacher leads the students in the reflection phase. The purpose of this phase is to encourage students to reflect on what they have learned in the previous two phases. Students are encouraged to analyze their errors (including the source of these errors) thereby uncovering misconceptions.

There are several theories that make framing a plausible teaching strategy. Cognitive Load Theory [3] suggests that problem solving for novices generates heavy working memory loads, which could be detrimental to learning. To balance these loads, teachers should

provide guided instruction, help narrow the problem search space by creating "boundaries" to each session, and alleviate the working memory restriction by making sure that only items relevant to the task are loaded into the working memory. This is exactly what happens during the priming phase.

Meta communication about the presentation of the subject is important for learning [4]. Prior to the learning activity, the student needs to know the boundaries of the lesson segment (exactly what the lesson will contain), which should be well defined by the teacher. The student needs to know the content of the session, differentiating between the old and the new material. They also need to know the links between the new knowledge and previously learned knowledge [4].

Many learning models view learning as a cyclic process, around which knowledge acquisition, knowledge application and reflection occur. Andreasen and Wu [5] discuss a few of the commonly used experiential models. Framing is a simplified (and thus possibly easier-to-implement) version of many of the models.

Reflection promotes deep learning [6, 7, 8]. Critically analysing the learning experience helps challenge the student's underlying perception of the domain, identify and correct misconceptions, and integrate new knowledge with existing knowledge [9]. This also allows the student to transfer the newly acquired knowledge to other types of problems or scenarios. Self-explaining one's actions [10] and monitoring one's progress via open student models [11, 12] have been shown to be useful reflective tools that benefit learning.

Our project consists of three main stages:

1. The learning activity is implemented within SQL-Tutor [2], while the pre- and post-action phases are facilitated by a human tutor. The purpose of this stage was to investigate the potential of framing to improve learning before actually implementing it in the ITS. The results of this stage were presented in [1] are reviewed in Section 1.
2. The pre-action and learning activity phases are implemented in SQL-Tutor. This is the current stage of our project.
3. The reflection phase is also implemented in the ITS.

We chose a suite of metrics early on to validate whether the way in which we implemented Framing helps to achieve the intent. These metrics include Learning Efficiency [13], help-usage metrics e.g. High-Level Help, Requests for Help [14, 15], meta-data about problems solved and problems attempted (including difficulty levels), learning curves, and pre and post-tests. The pre and post-tests were designed by a teacher to measure students' knowledge. The same pre and post-tests are to be used in all stages.

1. Stage 1: Simulating the Framing Strategy

As stated earlier, the purpose of this stage was to simulate the Framing strategy in the manner in which we planned to implement it in the ITS. This helped us gather some information about framing with regards to learning and test out our decisions prior to implementation. We selected a set of target SQL concepts, namely the concepts covered by queries using the *Group By* and *Having* clauses, which students generally find difficult to learn. SQL-Tutor was restricted to only present problems relating to these target concepts. The study was held immediately after the relevant concepts had been covered in lectures. The learning activity was problem solving in SQL-Tutor. The pre- and post-action phase were interactive, whiteboard, group sessions, led by a human teacher. The pre-action and post-action phases were limited to 10 minutes each, while the whole session lasted 100 minutes. In the pre-action phase, the teacher briefly reminded students of the target concepts (taught in lectures previously) and, eliciting student participation, worked through a few

examples of varying difficulty. The teacher also discussed typical misconceptions. After interacting with SQL-Tutor, the students were prompted to reflect on their learning experience by commenting on some of their own mistakes. The teacher also showed them the most common mistakes that are usually made during the problem-solving phase. Students were asked to find the errors (in terms of concepts and methods) in those incorrect solutions before collectively working through to reach a correct solution.

Thirty-eight students from a second year database course participated in the evaluation for no monetary reward. We divided participants randomly into two groups: experimental and control. The idea was to perform the evaluation in a setting that was as close to the normal learning environment faced by students. As such, the experimental and control sessions were held during regular course lab sessions. Students in both groups completed a pre-test and a post-test, which were of comparable difficulty and contained three questions relating to the target concepts with the maximum mark of 12. After the pre-test, the experimental group went through priming, followed by problem-solving and reflection phases, which were run as described. In contrast, the control group entered the problem-solving phase immediately after the pre-test. The pre-test, post-test, and problem-solving phases for both groups were identical.

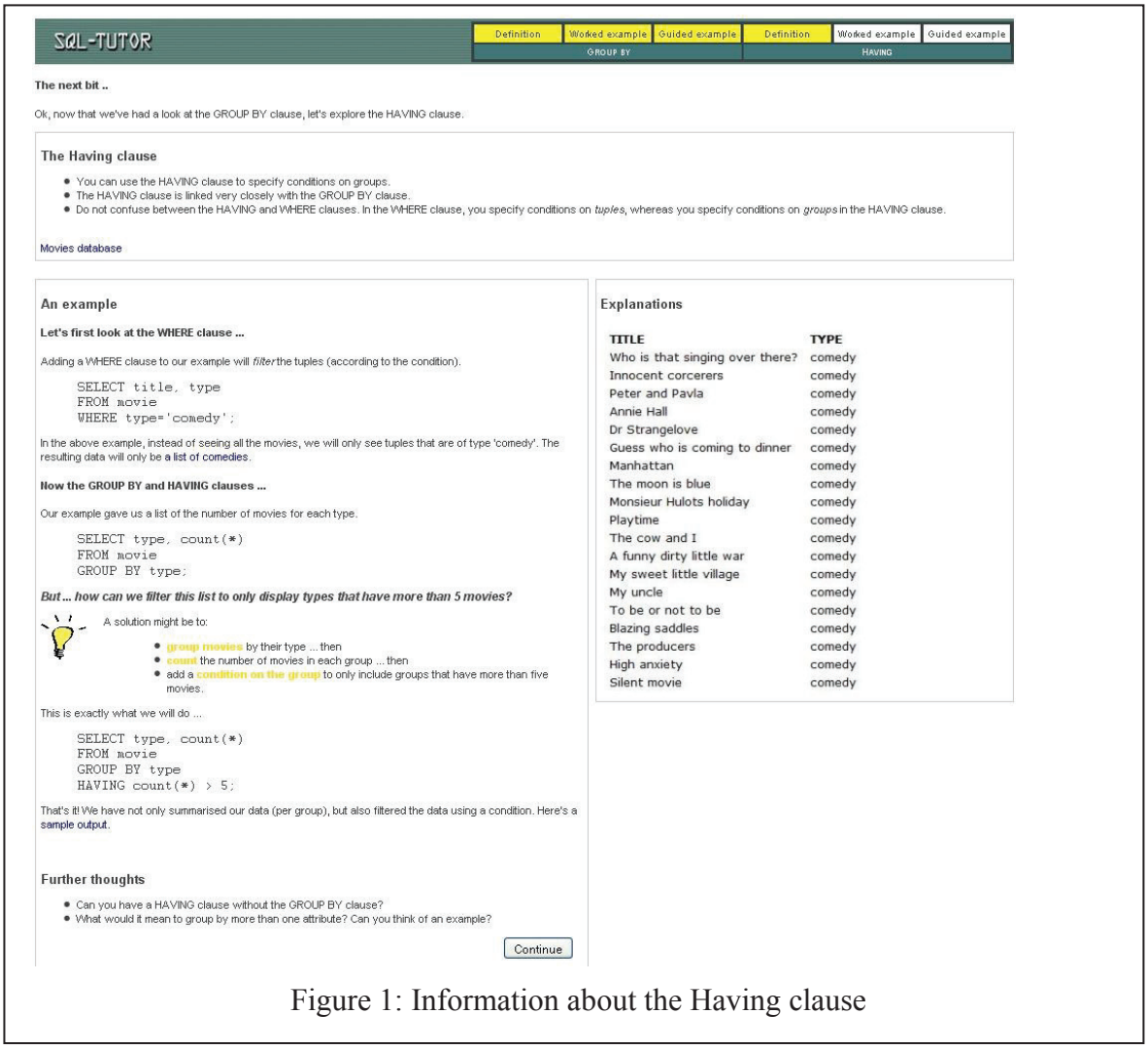

Figure 1: Information about the Having clause

The results of this preliminary study [1] showed that the experimental group had a higher problem-solving speed even though they attempted and solved problems of similar difficulty while using similar levels of help. Furthermore, the experimental group was significantly more efficient in their problem-solving phase than the control group. In other

words, while they did not learn more than the control group, they expended significantly less effort and therefore were more efficient.

2. Implementing Priming

We implemented the priming (pre-action) phase within the ITS using the lessons learned from stage 1. The design of this stage was similar, except that we excluded the post-action phase. The pre-action phase contained three steps for each of the target clauses (i.e. three for the Group By clause followed by three for Having clause). Each of the three steps increased from passive to more active in terms of student interaction. The first step contained the declarative knowledge about the clause followed by an example (see Figure 1). The example provided detailed explanation on how to solve the problem, and a possible solution. Students could also click on a link to display the result of the query. A "Further thoughts" section at the end gave more information to extend their knowledge of the clause. Once the student read the information on this page, they proceeded to the next step.

The second (and fifth) step contained a worked example. Students could hover over parts of the solution to get a detailed explanation for that part of the solution. Figure 2 shows the worked example and the explanation for the condition in the Having clause. Hovering over each part of the solution also highlighted the relevant part of the problem statement, allowing students to link the problem to the solution. Students could also click on a link to view the intended output of the query, or view the database schema.

The third (and sixth) step contained a strictly guided example. Similar to step 2, this step contained a problem statement and an empty solution statement (with blanks that the student had to fill in). When the student clicked on one of the blanks, the explanation for solving that part of the problem was displayed. Figure 3 shows the situation when the student asked for the explanation for the blank in the Having clause. The student could click the "Check" button to check their solution. If the student made an error on one of the parts of the solution, the explanation also contained a bottom out hint telling the student what to enter.

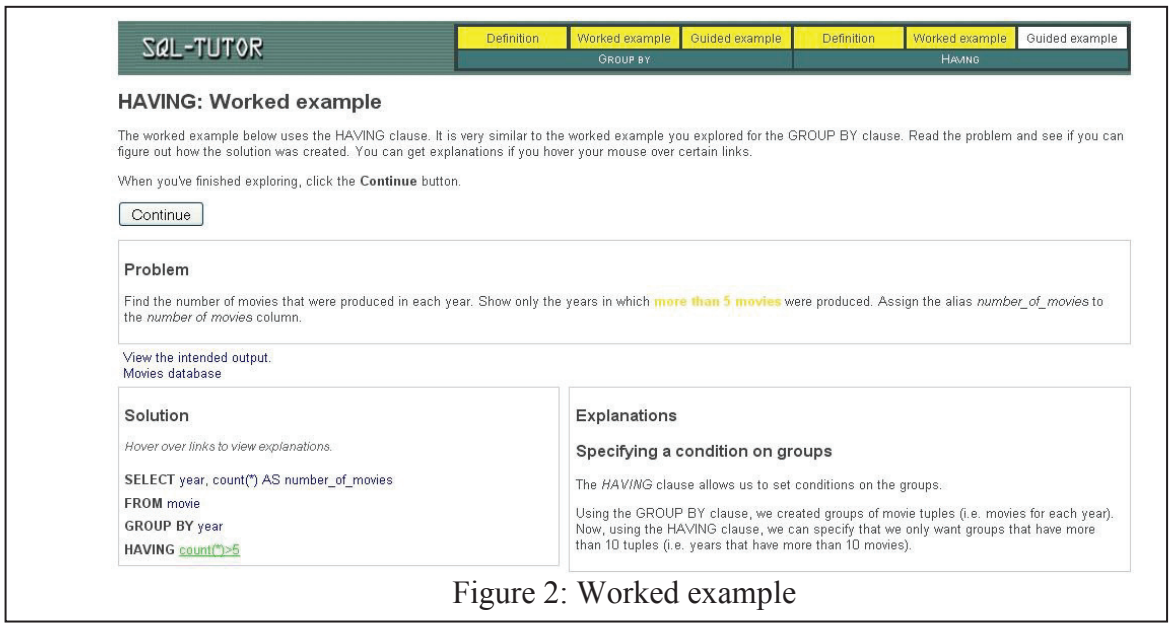

Figure 2: Worked example

The learning activity (problem-solving) immediately followed the six steps of the pre-action phase. This phase was identical to stage 1, where students worked on problems in SQL-Tutor. The problem set was restricted to problems using the target concepts.

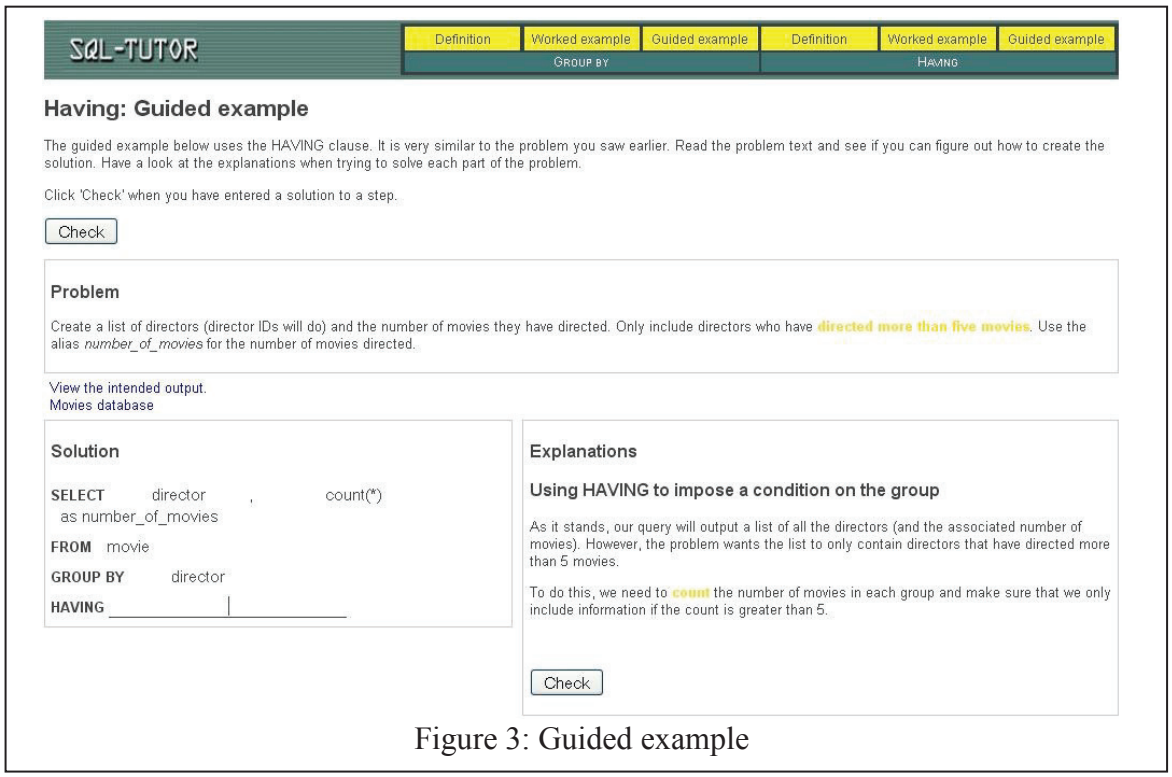

Figure 3: Guided example

3. Results

Thirty students participated in the evaluation for no monetary reward. We divided them randomly into two groups: experimental and control. Two sessions were held during regular course lab sessions (100 minutes long) on 13 and 14 May 2009 respectively. The students participated in the study during the lab session they normally attended throughout the course. Students in both groups completed the pre- and post-test (the same ones used in stage 1 of the project). Following the pre-test, the experimental group went through the pre-action phase in the newly added component, while the control group went directly onto the problem-solving phase.

Table 1: Matched means and standard deviations for test scores (%) and gains

	Pre-test	Post-test	Gain
Experimental group (*n*=5)	57.14% (*s.d* = 39.7)	75% (*s.d* = 16.6)	33.3% (*s.d* = 39.5)
Control group (*n*=12)	52.9% (*s.d* = 26.3)	88.3% (*s.d* = 11.2)	36.6% (*s.d* = 24.7)

The data we collected and analyzed included the pre/post-test results and just over 29 hours (total) of SQL-Tutor student models and logs in which 30 students collectively made 1,769 submissions to the system. There were 17 students in the control group and 13 in the experimental. However, only 17 students sat both tests, and we give the matched results in Table 1. There were no significant differences in the performances of the two groups on the pre-tests, post-tests or between gains (the gain is the difference between post- and pre-test score). There was a significant difference between the pre- and post-test performance of each group, indicating that students improved their domain knowledge during the session.

These results provide us with "trends" even with the low number of students from the experimental group that sat both tests (n=5).

The rest of the analyses were carried out on all the thirty students and the results are reported in Table 2. The trends in this stage were very similar to those found in stage 1. The experimental group spent less time solving problems; this was marginally significant ($t(25)=1.3$, $p=.09$). Students in both groups solved a similar number of problems. This means that the experimental group solved problems at a slightly faster rate, which was also marginally significant ($t(19)=0.46$, $p=.09$).

We analyzed the problem difficulty levels for both groups. Did students in one group attempt or solve problems that were significantly more difficult than the other that might account for the differing speed of problem solving? Each problem in SQL-Tutor is assigned a difficulty level by the SQL expert who authored the problems. Difficulty levels range from 1 (easy) to 9 (difficult) with non-trivial differences in difficulty between levels. SQL experts have checked problem difficulty levels such that problems with the same difficulty level are of similar difficulty. The problems attempted and solved were also of similar difficulty between groups. This was also confirmed for the highest and lowest difficultly level of problems attempted and solved in both groups i.e. students solved similar types of problems. On average, the experimental group made 49 (26.6) attempts while solving problems, while the control group made 68 (49.3) attempts; the difference was not significant showing that the both groups got similar amounts of feedback from the system. However, to check that students from one group did not receive higher levels of feedback (e.g. they used full solution much more than the other group), we calculated the high-level help used for both groups. *High-level help* (HLH) [14, 15] is defined as the type of help given by a system that provides (part or all of) the correct solution to the student rather than having the student to *solve* the problem; e.g. full solution is a type of HLH. Another important characteristic of HLH in SQL-Tutor is that the HLH levels have to be manually requested by the student whereas the ITS might automatically provide other types of feedback (Low-level help). The *HLH ratio* is the number of HLH attempts divided by the total number of attempts. This shows us the proportion of HLH use, from 0 (no HLH use) to 1 (the student used HLH on every attempt). Students from both groups used similar amounts of high-level help during this phase; 0.46 (0.26) for the experimental group and 0.43 (0.34) for the control group.

Table 2: Results for experimental and control groups

	Experimental	**Control**
Difficulty of problems attempted	5.02 (0.59)	4.95 (0.53)
Difficulty of problems solved	5.01 (0.55)	4.91 (0.56)
Lowest difficulty of problems attempted	3.53 (0.51)	3.64 (0.49)
Highest difficulty of problems attempted	7.0 (1.35)	6.82 (1.7)
Lowest difficulty of problems solved	3.61 (0.50)	3.64 (0.49)
Highest difficulty of problems solved	6.92 (1.32)	6.70 (1.82)
Number of problems solved	10.15 (5.03)	10.5 (6.4)
Time spent on problem solving (min)	52.46 (18.09)	65.17 (33.9)
High-level Help (HLH) ratio	0.46 (0.26)	0.43 (0.34)
Request for Help (RFH) attempts	1.84 (0.68)	1.88 (1.40)
Relative learning efficiency (E)	0.11	-0.12

The relative *learning efficiency* (E) is defined as the performance gained in one condition (the experimental condition) over the effort expended in relation to another condition (the control condition). A condition is more efficient if "1) their performance is higher than expected on the basis of their effort and/or 2) their invested effort is lower than might be expected on the basis of their performance" [13]. To calculate the efficiency of problem-solving for each group, we used "time" as the effort spent and "test gains" as the performance measure. The relative efficiency is found by first converting each of the raw

scores to a *z* score by subtracting the grand mean from the raw score and dividing by the standard deviation. E scores then are found by calculating the perpendicular distance between each *z* score and the E=0 line when plotted on a Cartesian graph. As with stage 1, the efficiency of the experimental group (E = 0.11) was higher than that of the control group (E = -0.12). This was marginally significant ($t(28)=1.11$, $p=0.1$, one-tailed, assuming unequal variances).

We also plotted learning curves for both groups (4). Although the differences were not significant, the trend lines indicate that the experimental group learned at a higher rate than the control group.

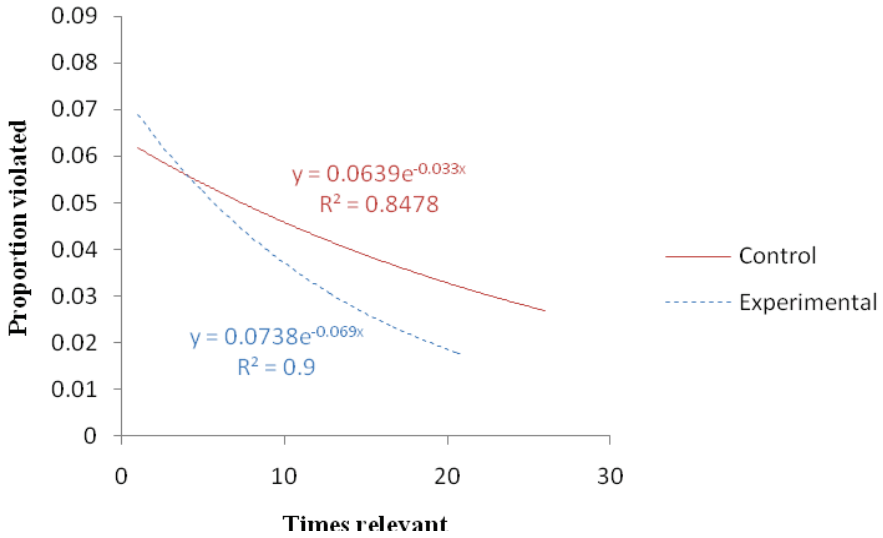

Figure 4: Case study 2: Learning curves for experimental and control groups

4. Discussion and Conclusions

This paper presented the second stage of our project, aiming to implement the framing teaching strategy in an ITS. In previous work, we performed a preliminary study with simulated framing, in which the pre-action and post-action stages were led by a human teachers instead of being implemented in the ITS. The aim of stage 1 was to see whether framing is an effective strategy for an ITS before actually implementing it and therefore committing significant recourses. The results of Stage 1 show that Framing results in significantly faster and more efficient learning.

In this current (second) stage of the project, we implemented the priming phase in SQL-Tutor. This is the first time framing has been implemented in ITSs. The trends gathered from the evaluation of this stage suggest that this implementation worked in a similar manner to that in stage 1. Note that this does not mean that we have achieved an ideal implementation. In fact, although the trends were similar to stage 1, the results gained were not as significant. We have pinpointed at least four possible reasons. First, we had a small number of participants, and therefore cannot make solid conclusions. Secondly, even though the pre-action phase in stage 1 was non-adaptive to the individual, the human teacher adapted to the group as a whole, especially during the worked examples step (when the teacher interacted with the group). This might have increased the effectiveness of the pre-action phase in stage 1. Thirdly, we decided to omit the "common misconceptions" step and only concentrate on correct knowledge. One reason was to keep the pre-action phase reasonably short (to stop it encroaching on the problem-solving). Another reason for the

omission was that presenting correct knowledge followed by incorrect knowledge (common misconceptions) did not seem intuitive using our method of presentation. Finally, the method of presentation differed in both stages. While we had a human teacher (animated, expressive, utilizing the student's visual and auditory senses) presenting in the first stage, we had a series of web pages with limited interaction in the second stage.

The results from this stage, added to that of the previous stage, increase our knowledge and give us a more detailed picture about various decisions we made and aspects of this strategy. Due to the evidence gathered in these stages, it is possible to implement the post-action phase in stage 3 and thus have a system that fully employs the Framing strategy in SQL-Tutor. However, information gathered from this stage suggests that we also could split the development path into a spike that evaluates some of the reasons given in the discussion above and tries to improve on the pre-action phase (say, stage 2B) while continuing development on stage 3. As we have gathered baseline information in stage 2 regarding the pre-action phase, the two stages (stage 2B and stage 3) can be undertaken concurrently. If the spike in stage 2B is successful, the improved pre-action phase can be added with confidence to the system at a later date.

References

[1] Mathews, M., Mitrovic, A. Does framing a problem-solving scenario influence learning? In: Kong, S.C., et al. (Eds.) Proc. 17th Int. Conf. Computers in Education ICCE 2009, Hong Kong: Asia-Pacific Society for Computers in Education, pp. 27-34, 2009.

[2] Mitrovic, A. (2003). An Intelligent SQL Tutor on the Web. *Artificial Intelligence in Education, 13*(2-4), 173-197.

[3] Kirschner, P. A., Clark, R. E., & Sweller, J. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist, 41*(2), 75-86.

[4] Leinhardt, G., & Ohlsson, S. (1990). Tutorials on the structure of tutoring from teachers. *Artificial Intelligence in Education, 2*(1), 21-46.

[5] Andreasen, R. J., & Wu, C.-H. (1999). Study Abroad Program as an Experiential, Capstone Course: a Proposed Model. 15th Annual Meeting of the Association for International Agricultural and Extension Education, Trinidad & Tobago.

[6] Katz, S., Allbritton, D., & Connelly, J. (2003). Going Beyond the Problem Given: How Human Tutors Use Post-Solution Discussions to Support Transfer. *International Journal of Artificial Intelligence in Education, 13*(1), 79-116.

[7] Katz, S., Connelly, J., & Wilson, C. (2007). Out of the Lab and into the Classroom: An Evaluation of Reflective Dialogue in Andes. In R. Luckin, K. R. Koedinger & J. E. Greer (Eds.), *Artificial Intelligence in Education: Frontiers in Artificial Intelligence and Applications* (Vol. 158, pp. 45-432): IOS Press.

[8] Lee, A. Y., & Hutchison, L. (1998). Improving Learning from Examples through Reflection. *Journal of Experimental Psychology: Applied, 4*(3), 187-210.

[9] Atkins, S., & Murphy, K. (1993). Reflection: A Review of the Literature. *Journal of Advanced Nursing, 18*(8), 1188-1192.

[10] Chi, M. T. H. (2000). Self-Explaining Expository Texts: The Dual Processes of Generating Inferences and Repairing Mental Models. In R. Glasser (Ed.), *Advances in Instructional Psychology: Educational Design and Cognitive Science* (Vol. 5, pp. 161 - 238). Mahwah, NJ: Lawrence Erlbaum Associates.

[11] Kay, J. (1997). *Learner Know Thyself: Student Models to give Learner Control and Responsibility.* Paper presented at the International Conference on Computers in Education, Sarawak, Malaysia.

[12] Mitrovic, A., & Martin, B. (2007). Evaluating the Effect of Open Student Models on Self-Assessment. *Artfiicial Intelligence in Education, 17*(2), 121-144.

[13] Paas, F. G. W. C., & Merrinboer, J. J. G. V. (1993). The Efficiency of Instructional Conditions: An Approach to Combine Mental Effort and Performance Measures. *Human Factors: The Journal of the Human Factors and Ergonomics Society, 35*, 737-743.

[14] Mathews, M., & Mitrovic, A. (2008). *How does students' help-seeking behaviour affect learning?* Paper presented at the 9th International Conference on Intelligent Tutoring Systems, Montreal, Canada.

[15] Mathews, M., Mitrovic, A. & Thomson, D. (2008). *How does students' help-seeking behaviour affect learning?* Analyzing high-level help seeking behaviour in ITSs. W. Nejdl et al. (Eds.): AH 2008, LNCS 5149, pp. 312-315, 2008.

References

[1] Ainsworth, S., and Grimshaw, S. Evaluating the REDEEM authoring tool: Can teachers create effective learning environments? *International Journal of Artificial Intelligence in Education 14*, 3,4 (Dec. 2004), 279–312.

[2] Ainsworth, S., Major, N., Grimshaw, S., Hayes, M., Underwood, J., Williams, B., and Wood, D. REDEEM: Simple intelligent tutoring systems from usable tools. *Tools for Advanced Technology Learning Environments* (2003), 205–232.

[3] Ainsworth, S., Underwood, J., and Grimshaw, S. Using an ITS authoring tool to explore educators' use of instructional strategies. In *Intelligent Tutoring Systems* (Berlin, Germany, 2000), G. Gauthier, C. Frasson, and K. VanLehn, Eds., Springer-Verlag Berlin Heidelberg, pp. 182–191.

[4] Ainsworth, S., Williams, B., and Wood, D. Using the REDEEM ITS authoring environment in naval training. In *Advanced Learning Technologies, 2001. Proceedings. IEEE International Conference on* (2001), pp. 189 –192.

[5] Akhras, F. N., and Self, J. A. System intelligence in constructivist learning. *International Journal of Artificial Intelligence in Education 11* (2000), 344–376.

[6] Aleven, V., Koedinger, K. R., and Cross, K. Tutoring answer explanation fosters learning with understanding. In *Artificial Intelligence in Education* (1999), IOS Press, pp. 199–206.

[7] Aleven, V., McLaren, B. M., Sewall, J., and Koedinger, K. R. The Cognitive Tutor Authoring Tools (CTAT): Preliminary

evaluation of efficiency gains. In *8th International Conference on Intelligent Tutoring Systems* (2006), M. Ikeda, K. Ashley, and T.-W. Chan, Eds.

[8] Aleven, V., Ogan, A., Popescu, O., Torrey, C., and Koedinger, K. Evaluating the effectiveness of a tutorial dialogue system for self-explanation. In *Intelligent Tutoring Systems* (Berlin, 2004), J. Lester, R. Vicari, and F. Paraguacu, Eds., Springer, pp. 443–454.

[9] Allen, V., and Feldman, R. *Studies on the role of tutor*. Academic Press, New York, USA, 1976, pp. 113–129.

[10] Anderson, J., and Reiser, B. The LISP tutor. *Byte 10*, 4 (1985), 159–175.

[11] Anderson, J., and Schunn, C. *Implications of the ACT-R Learning Theory: No Magic Bullets*, vol. 5. Erlbaum, Mahwah, NJ, 2000.

[12] Anderson, J. R. *Rules of the Mind*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.

[13] Anderson, J. R. ACT: A simple theory of complex cognition. *American Psychologist 51*, 4 (1996), 355 – 365.

[14] Anderson, J. R., Matessa, M., and Lebiere, C. ACT-R: a theory of higher level cognition and its relation to visual attention. *Hum.-Comput. Interact. 12*, 4 (Dec. 1997), 439–462.

[15] Andreasen, R. J., and Wu, C. H. Study abroad program as an experienctial, capstone course: A proposed model. In *The 15th Annual Meeting of the Association for International Agricultural and Extension Education* (Trinidad and Tobago, 1999).

[16] Aroyo, L., Dolog, P., Houben, G.-J., Kravcik, M., Naeve, A., Nilsson, M., and Wild, F. Interoperability in personalized adaptive learning. *Educational Technology & Society 9*, 2 (2006), 4–18.

[17] Atkins, S., and Murphy, K. Reflection: A review of the literature. *Journal of Advanced Nursing 18*, 8 (1993), 1188–1192.

[18] Baker, R., Corbett, A., Koedinger, K., Evenson, S., Roll, I., Wagner, A., Naim, M., Raspat, J., Baker, D., and Beck, J. Adapting to when students game an intelligent tutoring system. In *Intelligent Tutoring Systems*, M. Ikeda, K. Ashley, and T.-W. Chan, Eds., vol. 4053 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, pp. 392–401.

[19] Baker, R., Corbett, A., Roll, I., and Koedinger, K. Developing a generalizable detector of when students game the system. *User Modeling and User-Adapted Interaction 18*, 3 (2008), 287–314.

[20] Baker, R. S., Roll, I., Corbett, A. T., and Koedinger, K. R. Do performance goals lead students to game the system? In *Proceedings of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology* (Amsterdam, The Netherlands, The Netherlands, 2005), IOS Press, pp. 57–64.

[21] Baker, R. S. J. d., Corbett, A. T., Koedinger, K. R., and Roll, I. Detecting when students game the system, across tutor subjects and classroom cohorts. In *User Modeling 2005* (Berlin, Heidelberg, 2005).

[22] Baker, R. S. J. d., and Yacef, K. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining 1* (2009), 3–17.

[23] Baker, S., Mitrovic, A., and Mathews, M. Detecting gaming the system in constraint-based tutors. In *UMAP 2010* (Berlin Heidelberg, 2010), vol. 6075 of *LNCS*, Springer-Verlag, pp. 267–278.

[24] Bandura, A. Behavior theory and the models of man. *American Psychologist 29*, 12 (1974), 859 – 869.

[25] Bandura, A. Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review 84*, 2 (1977), 191 – 215.

[26] Bandura, A. *Self-regulation of motivation and action through internal standards and goal systems*. Erlbaum, Hillsdale, NJ, 1989, pp. 19–85.

[27] Bandura, A. A sociocognitive analysis of substance abuse: An agentic perspective. *Psychological Science (Wiley-Blackwell) 10*, 3 (1999), 214.

[28] Bandura, A., Grusec, J. E., and Menlove, F. L. Vicarious extinction of avoidance behavior. *Journal of Personality and Social Psychology 5*, 1 (1967), 16 – 23.

[29] Beck, J. Engagement tracing: using response times to model student disengagement. In *Artificial Intelligence in Education (AIED 2005)* (2005), pp. 88–95.

[30] Beck, J., Stern, M., and Haugsjaa, E. Applications of AI in education. *Crossroads 3* (September 1996), 11–15.

[31] Benzmuller, C., Fiedler, A., Gabsdil, M., Horacek, H., Kruijff-korbayova, I., Pinkal, M., Siekmann, J., Tsovaltzi, D., Vo, B. Q., and Wolska, M. A Wizard-of-Oz experiment for tutorial dialogues in mathematics. In *In Proceedings of the AIED Workshop on Advanced Technologies for Mathematics Education* (2003), pp. 471–481.

[32] Biswas, G., Jeong, H., Kinnebrew, J., Sulcer, B., and Roscoe, R. Measuring self-regulated learning skills through social interactions in a teachable agent environment. *Research and Practice in Technology-Enhanced Learning 5*, 2 (2010), 123–152.

[33] Biswas, G., Leelawong, K., Schwartz, D., Vye, N., and the TAG-V. Learning by teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence 19* (2005), 363–392.

[34] Biswas, G., Roscoe, R., Jeong, H., and Sulcer, B. Promoting self-regulated learning skills in agent-based learning environments. In *The 17th International Conference on Computers in Education* (2009), S. Kong, H. Ogata, H. Arnseth, C. Chan, T. Hirashima, F. Klett, J. Lee, C. Liu, C. Looi, M. Milrad, A. Mitrovic, K. Nakabayashi, S. Wong, and S. Yang, Eds., Asia-Pacific Society for Computers in Education, pp. 67–74.

[35] Bloom, B. S., Ed. *Taxonomy of Educational Objectives. Handbook I: Cognitive Domain.* Longmans, Green and Co., 1956.

[36] Bloom, B. S. The 2 sigma problem: The search for method of group instruction as effective as one-to-one tutoring. *Educational Researcher 13*, 6 (1984), 4–16.

[37] Brusilovsky, P. A framework for intelligent knowledge sequencing and task sequencing. In *Intelligent Tutoring Systems*, C. Frasson, G. Gauthier, and G. McCalla, Eds., vol. 608 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1992, pp. 499–506.

[38] Brusilovsky, P. Adaptive hypermedia. *User Modeling and User-Adapted Interaction 11* (2001), 87–110.

[39] Brusilovsky, P. KnowledgeTree: a distributed architecture for adaptive e-learning. In *The 13th international World Wide Web conference on Alternate track papers & posters* (New York, NY, USA, 2004), WWW Alt. '04, ACM, pp. 104–113.

[40] Brusilovsky, P., Mitrovic, A., Sosnovsky, S., Mathews, M., Yudelson, M., Lee, D., and Zadorozhny, V. Database exploratorium: a semantically integrated adaptive educational system.

In *Ubiquitous User Modeling Workshop at the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP 2009)* (2009).

[41] Brusilovsky, P., Sosnovsky, S., and Yudelson, M. Ontology-based framework for user model interoperability in distributed learning environments. In *World Conference on E-Learning, E-Learn 2005* (2005), AACE, pp. 2851–2855.

[42] Carbonell, J. R. AI in CAI: An Artificial Intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems 11*, 4 (1970), 190–202.

[43] Catrambone, R., and Yuasa, M. Acquisition of procedures: The effects of example elaborations and active learning exercises. *Learning and Instruction 16* (2006), 139–153.

[44] Chandler, P., and Sweller, J. Cognitive load theory and the format of instruction. *Cognition and Instruction 8*, 4 (1991), 293–332.

[45] Chandler, P., and Sweller, J. The split-attention effect as a factor in the design of instruction. *British Journal of Educational Psychology 62*, 2 (1992), 233–246.

[46] Chi, M., de Leeuw, N., Chiu, M., and LaVancher, C. Eliciting self-explanations improves understanding. *Cognitive Science 18* (1994), 439–477.

[47] Chi, M. T. H. *Self-Explaining Exository Texts: The Dual Processes of Generating Inferences and Repairing Mental Models*, vol. 5. Lawrence Erlbaum Associates, Mahwah, NJ, 2000, pp. 161–238.

[48] Clancey, W. J. Tutoring rules for guiding a case method dialogue. *IEEE Transactions on Man-Machine Systems 11*, 1 (1979), 25–49.

[49] Cohen, P., Kulik, J., and Kulik, C. Educational outcomes of tutoring: A meta-analysis of findings. *American Educational Research Journal 19*, 2 (1982), 237–248.

[50] Corbett, A. T., and Anderson, J. R. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction 4* (1995), 253–278.

[51] Cowan, N. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences 24* (2001), 87–114.

[52] Dalziel, J. Implementing learning design: The learning activity management system (LAMS), 2003.

[53] Dalziel, J. From reusable e-learning content to reusable learning designs: Lessons from LAMS. http://www.caudit.edu.au/educauseaustralasia/2005/PDF/B4.pdf, 2005.

[54] Dalziel, J. Lessons from LAMS for ims learning design. In *Sixth International Conference on Advanced Learning Technologies, 2006* (2006), IEEE, pp. 1101–1102.

[55] Dimitrova, V., Brna, P., and Self, J. The design and implementation of a graphical communication medium for interactive open learner modelling. In *International Conference on Intelligent Tutoring Systems* (Berlin Heidelberg, 2002), Springer Verlag, pp. 432–441.

[56] Dimitrova, V., Self, J., and Brna, P. The interactive maintenance of open learner models. In *The 9th International Conference of Artificial Intelligence in Education* (Amsterdam, 1999), S. Lajoie and M. Vivet, Eds., IOS Press, pp. 405–412.

[57] Dixon, N. *The organizational learning cycle. How we can learn collectively.* McGraw-Hill Book Company, London, 1994.

[58] Eiss, A. F., Harbeck, M. B., and Association, N. S. S. *Behavioral Objectives in the Affective Domain*. NEA Publications Sales, 1969.

[59] Ellerton, N. F. Children's made-up mathematics problems a new perspective on talented mathematicians. *Educational Studies in Mathematics 17* (1986), 261–271.

[60] Ericsson, K. A., and Charness, N. Expert performance: its structure and acquisition. *American Psychologist 49*, 8 (1994), 725–747.

[61] Ericsson, K. A., Krampe, R. T., and Tesch-Romer, C. The role of deliberate practice in the acquisition of expert performance. *Psychological Review 100*, 3 (1993), 363–406.

[62] Feng, M., and Heffernan, N. Informing teachers live about student learning: Reporting in the assistment system. *Technology Instruction Cognition AND Learning 3*, 1/2 (2006), 63.

[63] Freire, P. *Pedagogy of the Oppressed*. The Continuum Publishing Company, 1970.

[64] Fry, J., Ginzton, M., Peters, S., Clark, B., and Pon-Barry, H. Automated tutoring dialogues for training in shipboard damage control. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue - Volume 16* (Stroudsburg, PA, USA, 2001), SIGDIAL '01, Association for Computational Linguistics, pp. 1–4.

[65] Gakhal, I., and Bull, S. An open learner model for trainee pilots. *ALT-J: Research in Learning Technology 16*, 2 (2008), 123–135.

[66] Graesser, A., Person, N., and Magliano, J. Collaborative dialog patterns in naturalistic one-on-one tutoring. *Applied Cognitive Psychology 9* (1995), 359–387.

[67] Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W., and Harter, D. Intelligent tutoring systems with conversational dialogue. *AI Mag. 22*, 4 (Oct. 2001), 39–51.

[68] Greer, J., and McCalla, G. *Student Modelling: The Key to Individualized Knowledge-Based Instruction*. Springer-Verlag, 1994, p. 383.

[69] Hartley, D., and Mitrovic, A. Supporting learning by opening the student model. In *Intelligent Tutoring Systems*, S. Cerri, G. Gouardares, and F. Paraguacu, Eds., vol. 2363 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2002, pp. 453–462.

[70] Hirashima, T., Nakano, A., and Takeuchi, A. A diagnosis function of arithmetical word problems for learning by problem posing. In *PRICAI 2000 Topics in Artificial Intelligence*, R. Mizoguchi and J. Slaney, Eds., vol. 1886 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2000, pp. 745–755.

[71] Hirashima, T., Yokoyama, T., Okamoto, M., and Takeuchi, A. An experimental use of learning environment for problem-posing as sentence-integration in arithmetical word problems. In *Intelligent Tutoring Systems*, B. Woolf, E. Ameur, R. Nkambou, and S. Lajoie, Eds., vol. 5091 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, pp. 687–689.

[72] Hirashima, T., Yokoyama, T., Okamoto, M., and Takeuchi, A. Long-term use of learning environment for problem-posing in arithmetical word problems. In *16th International Conference on Computers in Education (ICCE 2008)* (Taipei, Taiwan, 2008), T.-W. Chan, G. Biswas, F.-C. Chen, S. Chen, C. Chou, M. Jacobson, Kinshuk, F. Klett, C.-K. Looi, A. Mitrovic, R. Mizoguchi, K. Nakabayashi, P. Reimann, D. Suthers, S. Yang, and J.-C. Yang, Eds., pp. 817–824.

[73] Hirashima, T., Yokoyama, T., and Takeuchi, A. Learning by problem-posing as sentence-integration and experimental use. In *Arti-*

ficial Intelligence in Education, R. Luckin, K. R. Koedinger, and J. E. Greer, Eds. IOS Press, 2007, pp. 254–261.

[74] Hung, D. Theories of learning and computer-mediated instructional technologies. *Educational Media International 38*, 4 (2001), 281–287.

[75] Hunt, D. E. *Beginning with ourselves in practice, theory, and human affairs*. Brookline Books, Cambridge MA, 1987.

[76] IMS-LD. Ims learning design faq. http://hdl.handle.net/1820/116, 2004.

[77] International Working Group on Educational Data Mining. Educational data mining. http://www.educationaldatamining.org/, 2009.

[78] Jackson, G. T., Ventura, M., Chewle, P., Graesser, A., and Group, t. T. R. The impact of Why/AutoTutor on learning and retention of conceptual physics. In *Intelligent Tutoring Systems*, J. C. Lester, R. M. Vicari, and F. Paraguau, Eds., vol. 3220 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2004, pp. 9–19.

[79] Jarvis, P. *Adult Learning in the Social Context*. Croom Helm, London, 1987.

[80] Jeong, H., Gupta, A., Roscoe, R., Wagster, J., Biswas, G., and Schwartz, D. Using hidden markov models to characterize student behaviors in learning-by-teaching. In *Intelligent Tutoring Systems*, vol. 5091 of *Lecture Notes in Computer Science*. 2008, pp. 614–625.

[81] Jordan, P., Makatchev, M., Pappuswamy, U., Vanlehn, K., and Albacete, P. A natural language tutorial dialogue system for physics. In *FLAIRS 2006* (Menlo Park, California, 2006), G. Sutcliffe and R. Goebel, Eds., AAAI Press, pp. 521–526.

[82] Kalyuga, S. Expertise reversal effect and its implications for learner-tailored instruction. *Educational Psychology Review 19* (2007), 509–539.

[83] Kalyuga, S., Ayres, P., Chandler, P., and Sweller, J. The expertise reversal effect. *Educational Psychologist 38* (2003), 23–31.

[84] Kasai, T., Nagano, K., and Mizoguchi, R. Building an ontology-based system which supports the instructional design process. In *The 18th International Conference on Computers in Education* (2010), S. L. WONG, S. C. KONG, F.-Y. YU, H. C. ARNSETH, B. CHANG, Y. S. CHEE, W. CHEN, T. HIRASHIMA, R. Y.-M. HUANG, F. KLETT, I. LEE, A. MITROVIC, K. NAKABAYASHI, H. OGATA, J. OSHIMA, P. REIMANN, M. SPECHT, T. SUPNITHI, T. TEO, and M. M. WANG, Eds., Asia-Pacific Society for Computers in Education, pp. 51–55.

[85] Katz, S., Allbritton, D., and Connelly, J. Out of the lab and into the classroom: An evaluation of reflective dialogue in andes. In *Artificial Intelligence in Education: Frontiers in Artificial Intelligence and Applications*, vol. 158. IOS Press, 2007.

[86] Katz, S., and Connelly, J. Going beyond the problem given: How human tutors use post-solution discussions to support transfer. *International Journal of Artificial Intelligence in Education 13*, 1 (2003), 79–116.

[87] Kay, J. Learner know thyself: Student models to give learner control and responsibility. In *The International Conference on Computers in Education* (Sarawak, Malaysia, 1997).

[88] Kay, J. Stereotypes, student models and scrutability. In *Intelligent Tutoring Systems*, G. Gauthier, C. Frasson, and K. VanLehn, Eds., vol. 1839 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2000, pp. 19–30.

[89] Kay, J., and Li, L. Scrutable learner modelling and learner reflection in student self-assessment. In *Intelligent Tutoring Systems*, M. Ikeda, K. Ashley, and T.-W. Chan, Eds., vol. 4053 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, pp. 763–765.

[90] Kelly, C. David Kolb, The Theory of Experiential Learning and ESL. *The Internet TESL Journal 3*, 9 (1997).

[91] Kirschner, P. A., Clark, R. E., and Sweller, J. Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist 41*, 2 (2006), 75–86.

[92] Kolb, A. Y., and Kolb, D. A. Learning styles and learning spaces: Enhancing experiential learning in higher education. *Academy of Management Learning & Education 4*, 2 (2005), pp. 193–212.

[93] Kolb, D. *Experiential Learning. Experience as a source of Learning and Development*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[94] Kolb, D. A. Experiential learning theory and the learning style inventory: A reply to Freedman and Stumpf. *Academy of Management Review 6*, 2 (1981), 289–296.

[95] Krathwohl, D. R. A revision of Bloom's Taxonomy: An overview. *Theory into Practice 41*, 4 (2002), pp. 212–218.

[96] Krathwohl, D. R., and Anderson, L. W. Merlin C. Wittrock and the revision of Blooms Taxonomy. *Educational Psychologist 45*, 1 (2010), 64–65.

[97] Krathwohl, D. R., Bloom, B. S., and Masia, B. B. *Taxonomy of Educational Objects. Handbook II: Affective Domain*. David McKay Company, Inc, New York, 1964.

[98] Lee, A. Y., and Hutchinson, L. Improving learning from examples through reflection. *Journal of Experimental Psychology: Applied 4*, 3 (1998), 187–210.

[99] Leinhardt, G., and Ohlsson, S. Tutorials on the structure of tutoring from teachers. *Journal of Artificial Intelligence in Education 2*, 1 (1990), 21–46.

[100] Major, N., Ainsworth, S., and Wood, D. REDEEM: Exploiting symbiosis between psychology and authoring environments. *International Journal of Artificial Intelligence in Education 8* (1997), 317–340.

[101] Mathews, M., and Mitrovic, A. Investigating the effectiveness of problem templates on learning in Intelligent Tutoring Systems. In *Proceedings of the 13th International Conference of Artificial Intelligence in Education (AIED 2007)* (Los Angeles, USA, 2007), R. Luckin, K. Koedinger, and J. Greer, Eds., pp. 611–613.

[102] Mathews, M., and Mitrovic, A. Do students who see more concepts in an ITS learn more? In *1st International Conference on Educational Data Mining* (Montreal, Quebec, Canada, 2008), R. S. J. d. Baker, T. Barnes, and J. E. Beck, Eds., Educational Data Mining, pp. 266–273.

[103] Mathews, M., and Mitrovic, A. How does students' help-seeking behaviour affect learning? In *Intelligent Tutoring Systems*, B. Woolf, E. Aimeur, R. Nkambou, and S. Lajoie, Eds., vol. 5091 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, pp. 363–372.

[104] Mathews, M., and Mitrovic, A. Does framing a problem-solving scenario influence learning? In *Proceedings of the 17th International Conference on Computers in Education* (2009), S. C. Kong, H. Ogata, H. C. Arnseth, C. K. K. Chan, T. Hirashima, F. Klett, J. H. M. Lee, C. C. Liu, C. K. Looi, M. Milrad, A. Mitrovic, K. Nakabayashi, S. L. Wong, and S. J. H. Yang, Eds., pp. 27–34.

[105] Mathews, M., and Mitrovic, A. Incorporating framing into SQL-Tutor. In *Workshop Proceedings of the 19th International Conference on Computers in Education* (2011), A. Mohd Ayub, B. Chang, and K. Leelawong, Eds., pp. 391–398.

[106] Mathews, M., Mitrovic, A., Lin, B., Holland, J., and Churcher, N. Do your eyes give it away? using eye tracking data to understand students attitudes towards open student model representations. In *Intelligent Tutoring Systems*, S. Cerri, W. Clancey, G. Papadourakis, and K. Panourgia, Eds., vol. 7315 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2012, pp. 422–427.

[107] Mathews, M., Mitrovic, A., and Thomson, D. Analysing high-level help-seeking behaviour in ITSs. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, W. Nejdl, J. Kay, P. Pu, and E. Herder, Eds., vol. 5149 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, pp. 312–315.

[108] McLaren, B. M., Lim, S. J., and Koedinger, K. R. When and how often should worked examples be given to students? New results and a summary of the current state of research. In *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (2008), pp. 2176–2181.

[109] Melis, E. Erroneous examples as a source of learning in mathematics. In *IADIS International Conference Cognition and Exploratory Learning in Digital Age 2004* (2004), pp. 311–318.

[110] Melis, E., Andres, E., Budenbender, J., Frischauf, A., Goguadze, G., Libbrecht, P., Pollet, M., and Ullrich, C. Activemath: A generic and adaptive web-based learning environment. *International Journal of Artificial Intelligence in Education 12* (2001), 385–407.

[111] Miettinen, R. The concept of experiential learning and John Dewey's theory of reflective thought and action. *International Journal of Lifelong Education 19*, 1 (2000), 54–72.

[112] Milik, N., Marshall, M., and Mitrovic, A. Teaching logical database design in ERM-Tutor. In *Proceedings of Intelligent Tutoring Systems 2006*, M. Ikeda, K. Ashley, and T.-W. Chan, Eds., vol. 4053 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, pp. 707–709.

[113] Miller, G. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review 63* (1956), 81–97.

[114] Mills, B., Evens, M., and Freedman, R. Implementing directed lines of reasoning in an intelligent tutoring system using the Atlas planning environment. *Information Technology: Coding and Computing, International Conference on 1* (2004), 729.

[115] Mitrovic, A. NORMIT: a web-enabled tutor for database normalization. In *Proceedings of the International Conference on Computers in Education 2002* (2002), vol. 2, pp. 1276–1280.

[116] Mitrovic, A. An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education 13*, 2-4 (2003), 173–197.

[117] Mitrovic, A., and Devedzic, V. A. A model of multitutor ontology-based learning environments. *Int. J. Continuing Engineering Education and Life-Long Learning. Special issue on ontologies. 14*, 3 (2004), 229–245.

[118] Mitrovic, A., Koedinger, K., and Martin, B. A comparative analysis of cognitive tutoring and constraint-based modeling. In *User Modeling 2003*, P. Brusilovsky, A. Corbett, and F. de Rosis, Eds., vol. 2702 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2003, pp. 147–147.

[119] Mitrovic, A., and Martin, B. Evaluating the effect of open student models on self-assessment. *International Journal of Artificial Intelligence in Education 19*, 2 (2007), 121–144.

[120] Mitrovic, A., Martin, B., and Suraweera, P. Intelligent tutors for all: The constraint-based approach. *IEEE Intelligent Systems 22* (2007), 38–45.

[121] Mitrovic, A., and Ohlsson, S. Constraint-based knowledge representation for individualized instruction. *Computer Science and Information Systems (COMSIS) 3*, 1 (2006), 1–22.

[122] Mitrovic, A., Ohlsson, S., and Barrow, D. The effect of positive feedback in a constraint-based intelligent tutoring system. *Computers & Education* (2012).

[123] Mitrovic, A., Suraweera, P., Martin, B., and Weerasinghe, A. DB-suite: Experiences with three intelligent, web-based database tutors. *Journal of Interactive Learning Research (JILR) 15*, 4 (2004), 409–432.

[124] Mitrovic, A., Suraweera, P., Martin, B., Zakharov, K., Milik, N., and Holland, J. Authoring constraint-based tutors in ASPIRE. In *Proceedings of the International Conference on Intelligent Tutoring Systems* (2006), M. Ikeda, K. Ashley, and T.-W. Chan, Eds., vol. 4053 of *LNCS*, pp. 391–398.

[125] Mitrovic, A., Williamson, C., Bebbington, A., Mathews, M., Suraweera, P., Martin, B., Thomson, D., and Holland, J. Thermo-Tutor: An intelligent tutoring system for thermodynamics. In *Global Engineering Education Conference (EDUCON), 2011 IEEE* (2011), pp. 378–385.

[126] Mizoguchi, R., Hayashi, Y., and Bourdeau, J. Ontology-based formal modeling of the pedagogical world: Tutor modeling. In *Advances in Intelligent Tutoring Systems*, R. Nkambou, J. Bourdeau, and

R. Mizoguchi, Eds., vol. 308 of *Studies in Computational Intelligence*. Springer Berlin / Heidelberg, 2010, pp. 229–247.

[127] Mohan, P., and Greer, J. E-learning specification in the context of instructional planning. In *Proceedings of AIED 2003: International Conference on Artificial Intelligence in Education* (2003), pp. 307–314.

[128] Mohan, P., Greer, J., and McCalla, G. Instructional planning with learning objects. In *IJCAI 2003 Workshop, Knowledge Representation and Automated Reasoning for E-Learning Systems* (2003), P. Baumgartner, P. Cairns, M. Kohlhase, and M. E., Eds., pp. 52–58.

[129] Moreno, R. When worked examples don't work: Is cognitive load theory at an impasse? *Learning and Instruction 16* (2006), 170–181.

[130] Murray, R., and vanLehn, K. Effects of dissuading unnecessary help requests while providing proactive help. In *Artificial Intelligence in Education* (2005), pp. 887–889.

[131] Murray, T. A model for distributed curriculum on the world wide web. *Journal of Interactive Media in Education 5* (1998), 1–32.

[132] Nakano, A., Hirashima, T., and Takeuchi, A. An evaluation of intelligent learning environment for problem posing. In *Intelligent Tutoring Systems*, S. Cerri, G. Gouarderes, and F. Paraguacu, Eds., vol. 2363 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2002, pp. 861–872.

[133] Nkambou, R., Belghith, K., and Kabanza, F. An approach to intelligent training on a robotic simulator using an innovative path-planner. In *Intelligent Tutoring Systems* (2006), Springer, pp. 645–654.

[134] Nkambou, R., Fournier-Viger, P., and Nguifo, E. M. Improving the behavior of intelligent tutoring agents with data mining. *IEEE Intelligent Systems 24*, 3 (May 2009), 46–53.

[135] Norrie, M., and Mitrovic, A. COSC366 Report: Combining Constraint-Based Tutors at a Conceptual Level. Tech. rep., University of Canterbury, Department of Computer Science & Software Engineering, 2012.

[136] Ohlsson, S. Constraint-based student modelling. *International Journal of Artificial Intelligence in Education 3*, 4 (1992), 429–447.

[137] Ohlsson, S. Learning from performance errors. *Psychological Review 103*, 2 (1996), 241–262.

[138] Ohlsson, S. *Computational Models of Skill Acquisition.* Cambridge, UK, 2008, pp. 359–395.

[139] Paas, F. G. W. C., and Merrinboer, J. J. G. V. The efficiency of instructional conditions: An approach to combine mental effort and performance measures. *Human Factors: The Journal of the Human Factors and Ergonomics Society 35* (1993), 737–743.

[140] Patel, A., Kinshuk, and Russell, D. Intelligent tutoring tools for cognitive skill acquisition in life long learning. *Educational Technology and Society 3*, 1 (2000).

[141] Polson, M. C., and Richardson, J. J., Eds. *Foundations of Intelligent Tutoring Systems.* L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1988.

[142] Prawat, R. S. Teachers' beliefs about teaching and learning: A constructivist perspective. *American Journal of Education 100*, 3 (1992), 354–395.

[143] Renkl, A., Atkinson, R. K., and Maier, U. H. From studying examples to solving problems: Fading worked-out solution steps helps learning. In *The Twenty-second Annual Conference of the Cognitive Science Society* (2000), L. R. Gleitman and A. K. Joshi, Eds., pp. 393–398.

[144] Resnick, L. B. *Constructing Knowledge in School*. Lawrence Erlbaum, Hillsdale, NJ, 1987, pp. 19–50.

[145] Rogers, A. *Teaching Adults*, 2 ed. Open University Press, Buckingham, 1996.

[146] Romero, C., and S., V. Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications 33* (2007), 135–146.

[147] Roscoe, R., and Chi, M. Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions. *Review of Educational Research 77*, 4 (2007), 534–574.

[148] Roscoe, R. D., Wagster, J., and Biswas, G. Using teachable agent feedback to support effective learning-by-teaching. In *The 30th Annual Meeting of the Cognitive Science Society* (2008), pp. 2381–2386.

[149] Rosé, C., Jordan, P., Ringenberg, M., Siler, S., VanLehn, K., and Weinstein, A. Interactive conceptual tutoring in atlas-andes. In *Proceedings of AI in Education 2001 Conference* (2001), pp. 151–153.

[150] Rourke, A., and Sweller, J. The worked-example effect using ill-defined problems: Learning to recognise designers' styles. *Learning and Instruction 19* (2009), 185–199.

[151] Rowley, K. The challenge of constructing a mega-tutor over the web. In *AIED 1997 Workshop on Intelligent Educational Systems on the World Wide Web* (Kobe, Japan, 1997), ISIR.

[152] Schulze, K., Shapiro, J., Shelby, R., Treacy, D., and Wintersgill, M. The andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education 15* (2005), 147–204.

[153] Seddon, G. M. The properties of Bloom's Taxonomy of Educational Objectives for the cognitive domain. *Review of Educational Research 48*, 2 (1978), pp. 303–323.

[154] Self, J. Bypassing the intractable problem of student modelling. *Intelligent tutoring systems: At the crossroads of artificial intelligence and education* (1990), 107–123.

[155] Silver, E., and Marshall, S. *Mathematical and scientific problem solving: Findings, issues and instructional implications*. Erlbaum, Hillsdale, NJ, 1989, pp. 265–290.

[156] Silver, E. A. On mathematical problem posing. *For the Learning of Mathematics 14*, 1 (1994), pp. 19–28.

[157] Silver, E. A., and Cai, J. An analysis of arithmetic problem posing by middle school students. *Journal for Research in Mathematics Education 27*, 5 (1996), 521–539.

[158] Silver, E. A., Mamona-Downs, J., Leung, S., and Kenney, P. Posing mathematical problems: An exploratory study. *Journal for Research in Mathematics Education 27*, 3 (1996), 293–309.

[159] Sosnovsky, S., Brusilovsky, P., Yudelson, M., Mitrovic, A., Mathews, M., and Kumar, A. Semantic integration of adaptive educational systems. In *Advances in Ubiquitous User Modelling*, T. Kuflik, S. Berkovsky, F. Carmagnola, D. Heckmann, and A. Krger, Eds., vol. 5830 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2009, pp. 134–158.

[160] Sosnovsky, S., Mitrovic, A., Lee, D., Brusilovsky, P., and Yudelson, M. Ontology-based integration of adaptive educational systems. In *16th International Conference on Computers in Education (ICCE 2008)* (Taipei, Taiwan, 2008), T.-W. Chan, G. Biswas, F.-C. Chen, S. Chen, C. Chou, M. Jacobson, Kinshuk, F. Klett, C.-K. Looi,

A. Mitrovic, R. Mizoguchi, K. Nakabayashi, P. Reimann, D. Suthers, S. Yang, and J.-C. Yang, Eds., pp. 11–18.

[161] Sosnovsky, S., Mitrovic, A., Lee, D., Brusilovsky, P., Yudelson, M., Brusilovsky, V., and Sharma, D. Towards integration of adaptive educational systems: mapping domain models to ontologies. In *6th International Workshop on Ontologies and Semantic Web for E-Learning (SWEL2008)* (Montreal, Canada, 2008), D. Dicheva, A. Harrer, and R. Mizoguchi, Eds.

[162] Stern, M., and Woolf, B. Curriculum sequencing in a web-based tutor. In *Intelligent Tutoring Systems*, B. Goettl, H. Halff, C. Redfield, and V. Shute, Eds., vol. 1452 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 1998, pp. 574–583.

[163] Stern, M., Woolf, B., and Kurose, J. Intelligence on the web? In *Artificial Intelligence in Education (1997)*. 1997, pp. 490–497.

[164] Su, W. M., and Osisek, P. J. The Revised Bloom's Taxonomy: Implications for Educating Nurses. *Journal of Continuing Education in Nursing 42*, 7 (2011), 321–327.

[165] Suraweera, P., and Mitrovic, A. An Intelligent Tutoring System for Entity Relationship modeling. *International Journal of Artificial Intelligence in Education 14*, 3–4 (2004), 375–417.

[166] Sweller, J. Cognitive load during problem solving: Effects on learning. *Cognitive Science 12*, 2 (1988), 257–285.

[167] Sweller, J. The worked example effect and human cognition. *Learning and Instruction 16*, 2 (2006), 165–169.

[168] Sweller, J., and Cooper, G. A. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction 2*, 1 (1985), pp. 59–89.

[169] Sweller, J., Van Merrinboer, J., and Paas, F. Cognitive architecture and instructional design. *Educational Psychology Review 10*, 3 (1998), 251–296.

[170] Thomson, D., and Mitrovic, A. Towards a negotiable student model for constraint-based itss. In *17th International Conference on Computers in Education (ICCE 2009)* (Hong Kong, 2009), S. Kong, H. Ogata, H. Arnseth, C. Chan, T. Hirashima, F. Klett, J. Lee, C. Liu, C. Looi, M. Milrad, A. Mitrovic, K. Nakabayashi, S. Wong, and S. Yang, Eds., Asia-Pacific Society for Computers in Education, pp. 83–90.

[171] Tsovaltzi, D., Melis, E., McLaren, B., Dietrich, M., Goguadze, G., and Meyer, A.-K. Erroneous examples: A preliminary investigation into learning benefits. In *Learning in the Synergy of Multiple Disciplines*, U. Cress, V. Dimitrova, and M. Specht, Eds., vol. 5794 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2009, pp. 688–693.

[172] Tsovaltzi, D., Melis, E., McLaren, B. M., Meyer, A.-K., Dietrich, M., and Goguadze, G. Learning from erroneous examples: when and how do students benefit from them? In *Proceedings of the 5th European conference on Technology enhanced learning conference on Sustaining TEL: from innovation to learning and practice* (Berlin, Heidelberg, 2010), EC-TEL'10, Springer-Verlag, pp. 357–373.

[173] Turner, T., Macasek, M., Nuzzo-Jones, G., Heffernan, N., and Koedinger, K. The assistment builder: A rapid development tool for its. In *Proceedings of the 12th Annual Conference on Artificial Intelligence in Education* (2005), pp. 929–931.

[174] VanLehn, K. *Student Modeling*. Erlbaum, Hillsdale, NJ, 1988, pp. 55–78.

[175] VanLehn, K. Cognitive skill acquisition. *Annual Review of Psychology 47* (1996), 513–539.

[176] VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M. The Andes Physics Tutoring System: Lessons learned. *International Journal of Artificial Intelligence in Education 15* (August 2005), 147–204.

[177] Vassileva, J., and Wasson, B. Instructional planning approaches: from tutoring towards free learning. *Proc. EuroAIED 96* (1996), 1–8.

[178] Vygotsky, L. *The Development of Higher Psychological Processes*. Harvard University Press, 1978.

[179] Ward, M., and Sweller, J. Structuring effective worked examples. *Cognition and Instruction 7*, 1 (1990), pp. 1–39.

[180] Weerasinghe, A., Mitrovic, A., and Martin, B. A preliminary study of a general model for supporting tutorial dialogues. In *16th International Conference on Computers in Education (ICCE2008)* (Taipei, Taiwan, 2008), T.-W. Chan, G. Biswas, F.-C. Chen, S. Chen, C. Chou, M. Jacobson, Kinshuk, F. Klett, C.-K. Looi, A. Mitrovic, R. Mizoguchi, K. Nakabayashi, P. Reimann, D. Suthers, S. Yang, and J.-C. Yang, Eds., pp. 125–132.

[181] Weerasinghe, A., Mitrovic, A., and Martin, B. Towards individualized dialogue support for ill-defined domains. *International Journal of Artificial Intelligence in Education 19*, 4 (2009), 357–379.

[182] Weerasinghe, A., Mitrovic, A., Thomson, D., Mogin, P., and Martin, B. Evaluating a general model of adaptive tutorial dialogues. In *Artificial Intelligence in Education*, G. Biswas, S. Bull, J. Kay, and A. Mitrovic, Eds., vol. 6738 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2011, pp. 394–402.

[183] Weil, S., and McGill, I. *Making sense of experiential learning: Diversity in theory and practice*. Open University Press, Milton Keynes, 1989.

[184] Wenger, E. Communities of practice; a brief introduction. http://www.ewenger.com/theory, 2006.

[185] Wenger, E., McDermott, R., and Snyder, W. *Cultivating communities of practice: a guide to managing knowledge*. Harvard Business School Press, 2002.

[186] Wenger, E., and Snyder, W. Communities of practice: the organizational frontier. *Harvard Business Review* (2000), 139–145.

[187] Wolf, B. *AI in Education*. John Wiley & Sons, Inc., New York, 1992, pp. 434–444.

[188] Yamamoto, S., Waki, H., and Hirashima, T. An approach to promote learning by problem-based problem-posing. In *Technology Enhanced Learning (TELearn 2009)* (2009), G.-J. Hwang, N.-S. Chen, and M.-P. Chen, Eds.

[189] Yudelson, M., Brusilovsky, P., Mitrovic, A., and Mathews, M. Using numeric optimization to refine semantic user model integration of educational systems. In *The 3rd International Conference on Educational Data Mining* (2010), pp. 221–230.

[190] zakharov, K., Ohlsson, S., and Mitrovic, A. Feedback microengineering in EER-Tutor. In *Artificial Intelligence in Education* (2005), C.-K. Looi, G. McCalla, B. Bredeweg, and J. Breuker, Eds., IOS Press, pp. 718–725.